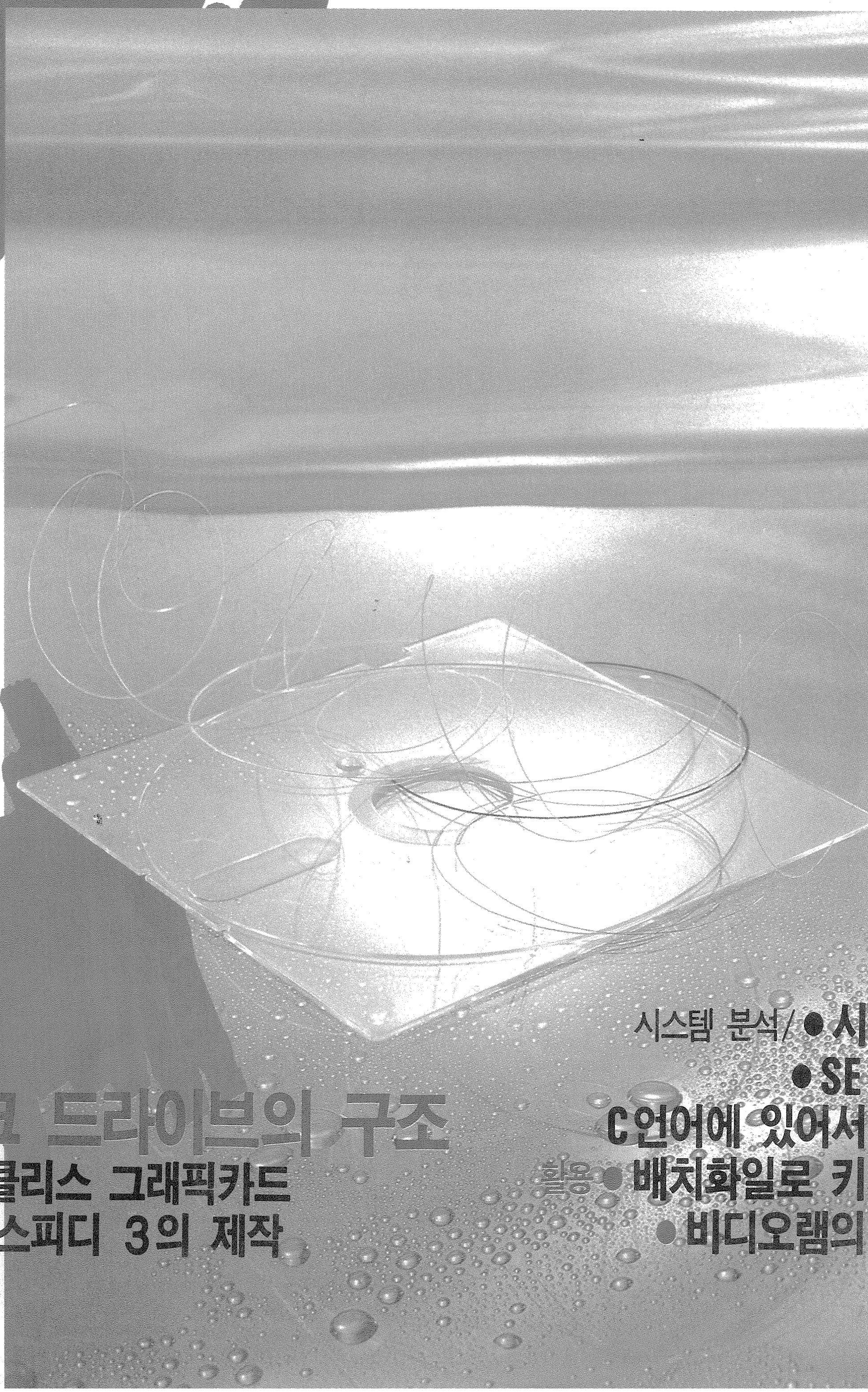
제2권 통권3호 1989년2월1일 발행(매월 1일 발행) 1988년10월10일 등록(등록번호 라-3794) ㊽140-150 서울특별시 용산구 갈월동 17

퍼스널컴퓨터

하이테크

89/02



특집/
디스크 드라이브의 구조

기획/ **허클리스 그래픽카드**

제작코너/ **스피디 3의 제작**

시스템 분석/ ● **시스템 자원**

● **SE I/O 포트**

C언어에 있어서의 최적화

활용 ● **배치화일로 키의 변환을**

● **비디오램의 속성변환**

□ 편집자노트 □

항상 조언과 격려를 바랍니다.

올 겨울은 유난히도 날씨가 변덕을 부립니다. 내내 따뜻한 영상의 날씨가 계속되다가도 하루 아침에 갑자기 영하의 날씨로 곤두박질하곤 합니다. 심술 많은 놀부의 심술보를 닮은 것처럼 변덕을 부려 많은 사람들이 이상난동에 고생을 하고 있읍니다. 독자여러분들도 감기가 걸리지 않게 조심하시기 바랍니다.
그러나 이러한 이상난동속에서도 항상 훈훈하게 느껴지는 곳이 있어 편집자의 마음은 흐뭇합니다. 전국곳곳에 산재한 컴퓨터 전문상가와 소프트하우스가 바로 그런 곳입니다. 방학이라 많은 학생들이 몰려나온 탓이려니 하기에는 이상현상이라고 생각됩니다.
또 최근들어 이러한 곳에 일반인들의 발길이 무척 잦다고 합니다. 컴퓨터가 일반화되고 대중화 되어가는 증세가 아닌가 생각됩니다.
우리 주위에 눈에 띄게 많아진 컴퓨터 때문에 많은 사람들이 불편을 하소연 하기도 합니다. 컴퓨터가 없으면 불편하기 때문에 컴퓨터를 들여놓았으나 제대로 사용할 줄을 몰라 고민하는 사람들과 이런 환경에서 컴퓨터를 사용할 줄 모르면 퇴보한다는 강박관념 때문에 서두르는 사람들도 무척 많습니다.
그런 이유때문인지는 몰라도 이번달에 들어온 300여통의 애독자 설문지를 보면 독자층이 학생보다는 일반인이 많으며 젊은층에 국한되지 않고 다양하게 분포되어 있읍니다. 그들 대부분은 컴퓨터를 다룬지 1년에서 3년 미만의 초보자가 대부분이며 새로 처음 시작하는 사람도 상당수입니다.
이들의 요구는 이구동성으로 보다 체계적으로 컴퓨터를 배울 수 있도록 쉽게 내용을 꾸며달라는 이야기 였읍니다. 저희들은 이들을 위하여 앞으로 보다 다양하고 쉽게 내용을 꾸미고자 합니다.
이번달에는 독자들이 가장 궁금히 여기는 디스크드라이브와 허큘리스카드를 특집과 기획으로 다루었읍니다. 보조기억장치로 컴퓨터를 사용하는데 반드시 필요한 디스크드라이브는 IBM-PC의 등장이후 이제 필수장비로 되어 있읍니다. 그러나 대부분의 사람들이 그 구조나 특성 등을 알지 못한채 단순히 사용하기만 하는 현실을 감안할 때 이번 기사는 디스크드라이브의 구조와 동작원리 등을 알게 해주어 많은 도움이 되리라 생각합니다. 또 기획은 허큘리스카드의 소프트 스위치 조작이나 제어방법 등에 관심을 가진 독자들에게 다양한 정보를 제공해 줄 것입니다.
최근 들어 일어나기 시작한 PC통신에 대한 붐은 많은 사람들에게 새로운 세계를 경험하게 해줍니다. 머지 않아 많은 사람이 이를 이용하리라 생각됩니다. 저희 잡지도 이들 관련된 자료를 정리하여 기사로 취급하고자 합니다.
이번달 부터는 애독자 여러분의 성원에 보답하기 위해 애독자 설문지를 수취인 요금후납부담으로 바꾸었읍니다. 많은 이용을 부탁드리며 항상 관심을 가지고 조언과 격려를 부탁드립니다. 작은 것일지라도 저희에게는 큰 힘이 되며 소중히 하여 책의 편집에 반영하겠읍니다.

MS-DOS 입문 [1]

IBM-PC의 첫걸음

박승제 /여주컴퓨터학원장

불과 몇년전만 하더라도 낯설게만 여겨졌던 컴퓨터라는 단어가 냉장고나 세탁기처럼 자연스럽게 들리며, 또 우리는 알게 모르게 컴퓨터와 더불어 생활을 하고 있다. 지하철의 자동 개찰, 은행의 온라인, TV의 그래픽 화면… 이제 우리는 바야흐로 컴퓨터시대에 살고 있으며, 컴퓨터는 우리 생활의 일부가 되어 가고 있다.

'사무실에 컴퓨터를 도입한다는 방침이 알려졌을 때 호기심 반 두려움 반의 심정으로 컴퓨터를 기대하였다. 그리고 컴퓨터가 사무실의 가장자리 책상위에 그 우아한(?) 자태를 드러내던 날 모두들 한두번씩은 키보드의 자판을 이것 저것 눌러보며 앞으로 이 괴물(?)을 어떻게 운용해야 하는가 하는 막막한 불안감에 사로 잡히기도 하였다.' 이 강좌는 이러한 경험을 하게 될 독자들을 위해서 마련한 것이므로 이제 컴퓨터를 배우려고 하는 독자들에게 충실한 길잡이가 되리라고 생각한다.

컴퓨터의 취급

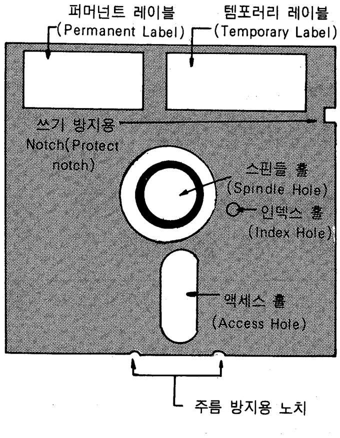
업무용 컴퓨터(IBM-PC)는 디스켓이라고 하는 플라스틱 조각에 각종 자료를 수록하게 된다. 디스켓은 디스크, 혹은 플로피디스크라고 부르기도 하는데 그것의 외부 명칭은 **그림 1**과 같다.

디스켓은 앞면과 뒷면이 구분되어 있으며 바르게 사용하지 않으면 자료를 수록하거나 읽을 수도 없으며 심하면 컴퓨터를 망가뜨리기도 한다. 이 디스켓에 자료를 써 넣거나 디스켓에서 읽어들이는 장치를 디스크 드라이브(보통 드라이브라고 부른다.) 라고 하는데 요즈음의 업무용 컴퓨터에는 드라이브가 한개 혹은 2개가 장착되어 있다.

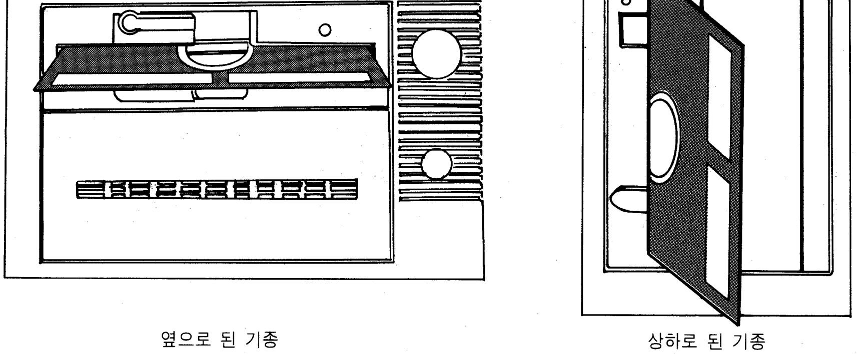
드라이브는 좌우로 혹은 상하로 긴 홈이 파져 있고 이 홈에 디스켓을 삽입하게 된다. 홈이 좌우로 파져있는 기종은 디스켓의 레이블이 붙어 있는 면이 위로 가도록 넣어야 하며, 홈이 상하로 되어 있는 기종의 경우에는 레이블이 좌측으로 가도록 디스켓을 삽입한다(**그림 2** 참조).

디스켓은 항상 레이블쪽을 손으로 잡아야 하며 액세스 홀을 손으로 잡아서는 안된다. 디스켓을 사용하지 않을 경우에는 항상 봉투에 넣어두도록 하고, 디스켓은 강한 자성물체나 습기, 극심한 온도 변화로 부터 격리시켜야 한다. 디스켓은 매우 민감하므로 디스켓을 컴퓨터의 모니터위에 방치하는 것 만으로도 디스켓내의 자료를 손상시킬 수 있다.

〔그림 1〕 디스켓의 외부 명칭



〔그림 2〕 디스켓의 삽입방법



옆으로 된 기종

상하로 된 기종

요즈음 업무용 컴퓨터는 하드디스크라고 하는 내장형 디스크드라이브를 장착하는 경우가 많은데 이 하드디스크는 매우 정교하게 제작이 되어 있으며, 충격에 약하므로 하드디스크가 내장된 컴퓨터를 이동할 경우에는 특히 주의하여야 한다. 하드디스크의 경우는 한번 설치하면 반영구적으로 사용하게 되므로 사용자가 제거하는 경우는 거의 없다.

컴퓨터에 여러대의 드라이브가 장착되어 있는 경우에는 A:, B:, C: 등과 같이 드라이브를 구분하다. 하드디스크는 항상 C:를 배정받으며, 플로피디스크는 A:와 B:를 배정받는데 윗쪽(혹은 좌측)에 위치하는 드라이브가 A:가 된다.(A, B, C 다음에 ':'가 붙어있는 것에 주의하라.)

1개의 플로피디스크와 1개의 하드디스크를 장착한 기종에서는 하드디스크가 C:를, 플로피디스크가 A:와 B:를 겸하게 된다. 이 경우에는 플로피디스크 1개 밖에 없지만 논리적으로 구별하게 되며, 어떤 경우에는 A:가 되고, 또 어떤 경우에는 B:가 된다.

드라이브에는 길이가 3㎝쯤 되는 레버가 있는데 이것을 '도어'라고 부른다. 디스크에 정보를 읽거나 쓸 경우에는 반드시 드라이브의 도어가 닫혀져 있어야만 한다. 그러나 컴퓨터를 사용하지 않을 경우에는 도어를 열어놓는 것이 좋다. 이때는 드라이브에 디스켓이 들어있어도 되고 없어도 되지만 먼지 등이 드라이브내에 들어가는 것을 막기 위해서는 드라이브에 디스켓을 넣어두는 것이 좋다고 할 수 있다. 컴퓨터를 사용하지 않을 경우에는 드라이브의 도어를 열어두어야 하지만, 컴퓨터를 이동할 경우에는 드라이브의 헤드가 흔들리는 것을 방지하기 위해서는 도어를 닫은 후에 이동하도록 한다. 요즈음에 보급되는 드라이브는 드라이브 내에 디스켓이 없는 경우에는 도어가 닫히지 않도록 설계되어 있으므로 빈 드라이브의 도어를 억지로 닫을려고 하지 말아야 한다.

컴퓨터의 시동

A: 드라이브에 DOS 디스켓을 넣고 도어를 닫은 다음 전원을 켠다. 전원을 켤 경우에는 모니터부터 켠다. 전원을 켜면 컴퓨터는 '자기 진단 프로그램'에 의해 컴퓨터의 이상유무를 점검한 다음, 이상이 없으면 '날짜를 입력하라'는 메시지와 함께 날짜를 입력받게 된다. 날짜를 입력하고 싶지않으면 그냥 〈Enter〉 키를 치면 된다.

날짜의 입력방법은 미국식, 유럽식, 동양식의 3가지이며 컴퓨터에서 어느 방식을 따르느냐는 기종마다 다르지만 한글 DOS의 경우에는 동양식을, 영문 DOS의 경우에는 미국식으로 되어있는 것이 보통이다.

동양식 : 년-월-일(YY-MM-DD)
미국식 : 월-일-년(MM-DD-YY)
유럽식 : 일-월-년(DD-MM-YY)

YY-MM-DD 등의 메시지는 년, 월, 일을 각각 2자리이내로 입력하라는 의미이다. 만약 날짜의 입력이 제대로 되지 않았을 경우에는(2월 30일을 입력한 경우 등) 입력이 잘못 되었음을 표시한 후 다시 날짜를 입력받게 된다.

날짜의 입력이 끝나면 컴퓨터는 현재의 시각을 표시한 후 새로운 시각을 입력할 것을 요구한다. 시각을 변동하고 싶지않는 경우에는 그냥 〈Enter〉를 치면 되고, 변경할 경우에는 '시:분:초'의 형식으로 입력한다. 이어 컴퓨터는 화면 1과 같은 메시지를 출력한 다음 A>를 표시하고 사용자의 명령을 받아들이게 된다. 컴퓨터는 이제 우리의 명령을 받아들일 준비가 다 된 셈이다. 이것을 다른 말로 표현하면 'DOS를 부팅했다'라고 한다. 즉, 컴퓨터는 디스켓으로부터 DOS 프로그램을 읽어들여 실행한 것이다.

DOS란 무엇인가?

인간에게는 인간 고유의 언어가 있듯이 컴퓨터에게는 컴퓨터가 이해할 수 있는 언어가 있다. (언어학적인 측면에서 보면 언어라고 해도 별로 나무랄 일도 아니니다.) 언어랄 것도 없지만, 잡다한 기호 따위로 표기하고 있으므로 DOS는 인간의 말을 컴퓨터가 알 수 있는 말로 번역하여 컴퓨터로 하여금 인간의 명령을 수행하도록 하는 일종의 프로그램이다. 컴퓨터를 호텔에 비유한다면 DOS는 그 호텔의 지배인쯤 되는 셈이다. 컴퓨터를 효율적으로 운용하려면 좋던 싫던 DOS를 이해하고, 또 DOS와 친해져야만 한다.

최초의 명령

우선 CLS라고 입력한다. 대문자도 좋고 소문자라도 괜찮다. DOS는 소문

〔화면 1〕 DOS의 초기화면

```
Current date is Tue  1-01-1980
Enter new date (mm-dd-yy): 1-25-89
Current time is  0:00:44.76
Enter new time: 12:35:12


The IBM Personal Computer DOS
Version 3.30 (C)Copyright International Business Machines Corp 1981, 1987
             (C)Copyright Microsoft Corp 1981, 1986

A>_
```

자로 입력해도 대문자로 기억한다. 입력 중 잘못된 글자는 〈Back Space〉 키를 눌러서 지울 수도 있다. 입력을 했으면 〈Enter〉 키를 쳐라. 컴퓨터의 화면이 깨끗이 지워졌을 것이다.

〈Enter〉 키는 입력이 끝났음을 알려주는 키로 모든 명령은 〈Enter〉 키를 입력해 주어야만 비로소 수행하게 된다. 군대 용어로 치면 '실시!'에 해당한다고 볼 수 있다. 이제부터는 〈Enter〉 키는 '↵'으로 표시하기로 한다.

CLS ↵ 는 CLS를 입력하고 〈Enter〉 키를 치라는 의미이다.

CLS는 Clear Screen의 약어로 화면을 소거하라는 명령이다.

다음에는 DIR ↵ 을 입력해보자.

드라이브에 불이 켜지며 화면 2와 같은 내용이 표시되었을 것이다.

DIR은 디스크에 수록되어 있는 자료의 제목을 화면에 출력하라는 명령이다. 컴퓨터는 자료를 화일(File)이라는 단위로 저장한다. 화일은 디스크에 수록하는 자료의 단위로 그 크기는 작업하는 양에 따라 각각 다르기 마련이다. 어떤 사람은 기안용지의 한장 분을 수록해 두는 경우도 있겠고, 또 어떤 사람은 수천명 분의 주소를 수록해 두는 경우도 있을 수 있다. 컴퓨터는 이들의 자료를 각각 독립된 자료로 관리 하게 되며, 이들 하나 하나를 화일이라고 부른다.

화일들은 각각 고유한 이름을 부여하여야 하는데 화일에 이름을 붙이는 데는 몇가지 규칙이 있다.

화일이름 붙이기

화일이름은 8자 이내의 공백과 '.'을 포함하지 않는 문자열로 구성하여야 하며, 소문자로 입력해도 대문자로 바뀌게 된다. 한글 DOS의 경우에는 화일이름을 한글로 부여할 수도 있다. 그러나 영문 DOS의 경우에는 한글이름을 부여하면 나중에 이상한 문자로 바뀌게 되므로 사용하지 않는 것이 좋다.

한글 DOS란 화면상의 메시지가 한글로 표시되는 DOS를 말한다. 즉, '날짜를 입력하십시요'처럼 표시되는 것은 한글 DOS요, 'Enter new date'로 표시되는 것은 영문 DOS이다. 한글이거나 영문이거나 화면상의 메시지만 다를 뿐 명령어는 모두 영어로 내리게 되므로 동일한 것으로 취급한다.

사람의 경우에는 이름과 성이 있듯이 화일도 사람의 성에 해당하는 확장자명(Extention)이라는 것을 붙일 수 있다. 확장자명은 3자 이내의 문자열로 구성되며 화일명의 경우처럼 공백과 '.'를 포함하면 안된다.

화일명에 확장자명을 부여할 경우에는 화일과 확장자명 사이를 '.'로 구분한다. 화면 2에 나와있는 화일들을 예로 들면 COMMAND.COM, ANSI.SYS와 같이 입력하여야 한다. 화면 2에는 화일명과 확장자명 사이를 공백으로 분리하여 표시하고 있지만 우리는 반드시 '.'으로 구분하여야 한다. 화일명이나 확장자명은 컴퓨터의 사용자 자신이 임의로 부여하게 되지만 특별한 용도로 예약이 되어 사용자가 사용하면 안될 것이 몇 개 있다. 이것들을 나열하면 다음과 같다. SYS, COM, EXE, BAT, BAK, BAS, COB, FOR, C, PAS, ASM, OBJ…

SYS는 그 화일이 시스템용 화일임을 나타낸다. 즉, 컴퓨터가 처음 켜질때 스스로 읽어들여서 실행하는 화일로서 사용자가 관여하지 않도록 되어 있다. 단, CONFIG.SYS라는 화일만은 예외로 사용자가 작성하는 것인데 이것에 대해서는 차후에 배우기로 한다.

COM, EXE, BAT는 DOS상태에서 실행할 수 있는 화일로서 프로그래머들이 작성해 둔 응용 프로그램들이다. 그것들의 기능은 프로그램에 따라 각각 다르다. 실제로 우리는 이런 프로그램들을 DOS상에서 실행함으로써 비로

〔화면 2〕 DIR의 실행 결과

```
A:\>dir

 Volume in drive A has no label
 Directory of  A:\

ASSIGN   COM     1561   3-17-87  12:00p
BACKUP   COM    31913   3-18-87  12:00p
BASIC    COM     1063   3-17-87  12:00p
CHKDSK   COM     9850   3-18-87  12:00p
CLOUD    COM     8048   1-04-80  11:33p
COMMAND  COM    25307   3-17-87  12:00p
COMP     COM     4214   3-17-87  12:00p
DEBUG    COM    15897   3-17-87  12:00p
DISKCOMP COM     5879   3-17-87  12:00p
DISKCOPY COM     6295   3-17-87  12:00p
ANSI     SYS     1678   3-17-87  12:00p
COUNTRY  SYS    11285   3-17-87  12:00p
DISPLAY  SYS    11290   3-17-87  12:00p
DRIVER   SYS     1196   3-17-87  12:00p
APPEND   EXE     5825   3-17-87  12:00p
ATTRIB   EXE     9529   3-17-87  12:00p
BASICA   EXE    54272   5-13-83  12:00p
       17 File(s)    147456 bytes free
A:\>
```

소 컴퓨터의 진가를 이용할 수 있게 된다. DOS는 이들을 실행시키는 하나의 도구일 뿐이다.

BAK는 백업용 화일로서 프로그램을 실행하는 과정에서 생기는 화일을 만약을 대비하여 복사해둔 화일로 사용자가 복사하는 것이 아니라 응용 프로그램이 처리해준 것이다. BAS, COB, FOR… 등의 확장자명은 프로그램 언어로 작성된 프로그램임을 나타내는데 이것들은 DOS에서 실행이 불가능한 화일들이다.

참고로 기술하자면 BAS는 베이직 언어로, COB는 코볼, FOR는 포트란, C는 C언어, PAS는 파스칼, ASM은 어셈블리, OBJ는 기계어로 작성되어 있음을 나타낸다. BAS, COB 등의 확장자명은 절대 사용금지된 확장자명은 아니지만 다른 사람들이 프로그램으로 오해할 소지가 있으므로 가능하다면 사용하지 않는 것이 좋다.

DOS에서 우리가 실행할 수 있는 화일들은 확장자명이 COM, EXE, BAT인 것들 뿐이다.

화일목록의 표시

위에서 우리는 DIR이란 명령이 화일의 제목을 표시한다는 것을 알았다. 그러면 DIR이라는 명령에 대해 좀 더 알아보기로 하자.

책에는 차례가 있어서 원하는 부분의 페이지를 쉽게 찾아볼 수 있게 해주듯이 디스켓에도 그 속에 수록된 화일에 관한 정보를 모아두는 부분이 있다. 이곳을 컴퓨터용어로 디렉토리(Directory)라고 하는데 DIR은 이 디렉토리를 화면에 표시하라는 명령이다.

다시 한 번 DIR을 입력해보자. 화일의 수가 많다보니 처음의 내용을 미처 보기도 전에 화면위로 사라져 버린다. 화면의 표시를 일시 중지하고 싶을 경우에는 〈Ctrl〉 키를 누른 상태에서 〈S〉를 누른다. 이것을 우리는 〈Ctrl〉+〈S〉로 표시하기로 하자. 〈Ctrl〉+〈S〉는 '콘트롤 S' 라고 읽는다.

〈Ctrl〉+〈S〉는 화면의 출력을 일시 중단하는 키로 다시 출력을 계속할 경우에는 〈Ctrl〉+〈C〉를 제외한 아무키나 누르면 된다.

〈Ctrl〉+〈C〉는 자신이 내린 명령을 중도에 취소하는 경우에 쓰인다. 〈Ctrl〉+〈C〉를 입력하면 컴퓨터는 하던 일을 그 시점에서 종료하게 된다.

〈Ctrl〉+〈S〉와 비슷한 키로는 〈Ctrl〉+〈Num Lock〉키가 있으며, 〈Ctrl〉+〈Scroll Lock〉키는 〈Ctrl〉+〈C〉와 동일한 기능을 한다. 이런 기능키들의 사용법을 익혀두면 나중에 유용하게 쓰일 때가 있을 것이다. 〈Ctrl〉+〈S〉로 화면을 일시 정지하면서 화일목록을 보는 것은 상당히 불편하다.

이번엔 DIR / P ↵ 라고 입력해보자.

DIR / P는 디렉토리를 화면(페이지) 단위로 출력하라는 명령이다. 컴퓨터는 디렉토리를 표시하다 화면에 가득 차게 되면 스스로 출력을 멈추고 '아무키나 입력하라'는 메시지를 표시하게 된다. 다음 화면을 보려면 〈Ctrl〉+〈C〉를 제외한 아무키나 누르면 된다.

이번에는 DIR / W ↵ 를 입력해보라. DIR / W는 디렉토리를 넓게(Wide) 표시하라는 명령으로 화일 이름 뒤에 표시되는 크기, 날짜, 시간 등을 제거하고 한줄에 여러개의 화일명을 표시한다.

와일드카드 문자

이번엔 확장자명이 COM인 화일만 표시해보자. DIR *.COM 을 입력해본다. (DIR 다음에 한 칸 띄운 것에 주의하라) 화면에는 확장자명이 COM인 화일목록만 출력하게 된다. DIR *.EXE 를 입력하면 확장자명이 EXE인 화일만 출력될 것이다.

이제 DIR C*.COM 을 입력해보자. 화면에는 화일명이 'C'로 시작되고 확장자명이 COM인 화일만 출력된다. '*'(Asterisk라고 읽는다)는 '그 이하의 문자열은 무엇이든지 좋다'라는 의미의 문자이다. 이를테면 'ABC*.*'는 화일명이 ABC로 시작하고, 확장자명은 무엇이든지 상관없는 화일들을 의미한다. '* .*'는 '화일명이 무엇이든지 또, 확장자명이 무엇이든지 좋다'라는 의미이므로 다시 말하면 모든 화일을 의미하게 되는 것이다.

이와 같은 문자로는 '?'가 있는데 '*'가 그 이하의 문자열을 대신해 주는 반면 '?'는 단 1개의 문자만을 대치해준다.

'KIM-?.DAT'는 'KIM-'으로 시작하고 그 다음의 1문자는 아무거나 좋으며, 확장자명은 'DAT'이다 라는 의미이므로 'KIM-0.DAT', 'KIM-K.DAT' 등은 여기에 포함된다. 그러나 'KIM-25.DAT'는 여기에 포함되지 않는다. '?'는 단 1자만을 대치하기 때문에 '?'다음의 '5'는 포함할 수 없게 된다. 이 때는 'KIM-??.DAT'로 하든지, 'KIM-*.DAT'로 지정하면 된다. 그러면 'K??-*.?A?'에는 어떤화일들이 포함될 수 있는지 각각 생각해 보라.

DOS에서는 '*'과 '?'를 와일드카드(Wild Card) 문자라고 부른다. 트럼프로 포커를 해 본 경험이 있는 독자들은 와일드카드의 의미가 금방 이해가 갈 것이다.

드라이브의 지정

지금까지는 A:드라이브의 내용이 표시되었다. 이번에는 다른 드라이브에 있는 디스켓의 내용을 표시해보자. 지금 화면에는 'A〉'라고 표시되어 있는 것을 주목하기 바란다. 'A〉'라는 의미는 현재 A:드라이브가 지정되어 있음을 표시하는 DOS의 메시지이다. 우리는 이것을 프롬프트(Prompt)라고 부른다. 프롬프트는 컴퓨터가 명령대기 상태임을 나타내주는 기호들을 총칭해서 부르는 말이다.

컴퓨터는 A:드라이브로부터 시작하려는 속성을 가지고 있으므로 DOS가

부팅된 다음에는 A:를 표시한다. 그러나 하드디스크로 부터 DOS를 부팅한 경우에는 C:를 표시하게 된다.

A〉는 현재 A:가 지정되어 있으며, 드라이브의 지정을 생략하면 A:를 지정하지 않을 때 DOS가 알아서 지정하는 드라이브를 커런트 드라이브라고 한다. 커런트 드라이브의 변경은 B:↵, C:↵, A:↵ 처럼 입력하면 된다.

B:드라이브의 내용을 표시할 경우에는 우선 B:↵를 입력한다. 그러면 화면에는 B〉가 나온다. 이것은 커런트 드라이브가 B:로 변경되었다는 신호이다. 이어 DIR 을 입력하라. 이제는 B:드라이브의 화일목록이 출력될 것이다.

커런트 드라이브를 변경하지 않고 다른 드라이브의 내용을 볼려면 DIR B:↵ 처럼 명령어 다음에 드라이브를 지정하면 된다.

그렇다면 다음의 명령은 무슨 내용인지 생각해보기 바란다.

```
DIR B:*.COM
DIR B:/P
DIR B:/W
DIR B:*.COM/P
DIR A:*.*/W
```

화면의 내용을 프린터로 출력

화면의 내용을 프린터로 출력하는 방법은 몇가지가 있다. 화일디렉토리의 내용을 프린터로 출력하는 경우를 예로 들기로 한다.DIR B:PRN 라고 입력해 보라. (키를 누르기 전에 프린터를 켜두어야 한다) 디렉토리의 내용이 화면에 출력되는 대신에 프린터로 찍힐 것이다. DOS에서 '〉'는 명령의 실행 결과를 그 다음의 장치에 출력하라는 의미로 쓰인다. 그렇다면 PRN은 프린터의 장치명임에 틀림없다.

'〉PRN'으로 프린터로 출력하는 경우 화면에는 아무 것도 출력되지 않고 프린터에만 출력이 된다. 실행 결과를 화면에도 표시하면서 프린터에도 출력하려면 〈Ctrl〉+〈P〉를 누른다. 〈Ctrl〉+〈P〉를 눌러도 화면에는 아무런 표시도 나타나지 않지만 그 키를 누른 시점부터 화면에 나타나는 모든 내용은 프린터로 출력된다. 프린터의 출력을 중지하려면 〈Ctrl〉+〈P〉를 다시 누르면 된다. 위의 내용을 화면 3에 나타냈으니 참고하기 바란다.

화면의 내용을 프린터에 출력하는 다른 방법은 화면을 하드카피(Hard Copy)하는 것이다. 하드 카피란 화면의 내용을 그대로 프린터에 출력하는 것을 의미하는 것으로 화면에 그림이 그려져 있다면 그 그림까지도 프린터에 복사된다. 화면의 하드 카피는 〈S-

〔화면 3〕 프린터 출력

```
A:\>dir  ←―――――――― <CTRL>+<P>를 누른다(화면에 나타나지 않음).

 Volume in drive A has no label
 Directory of  A:\

ASSIGN   COM     1561   3-17-87  12:00p  ┐
BACKUP   COM    31913   3-18-87  12:00p  │
BASIC    COM     1063   3-17-87  12:00p  │
CHKDSK   COM     9850   3-18-87  12:00p  │
CLOUD    COM     8048   1-04-80  11:33p  ├―― 화면과 프린터에 동시 출력된다.
COMMAND  COM    25307   3-17-87  12:00p  │
COMP     COM     4214   3-17-87  12:00p  │
DEBUG    COM    15897   3-17-87  12:00p  │
DISKCOMP COM     5879   3-17-87  12:00p  ┘
DISKC ←―― <CTRL>+<P>를 눌렀다.
A:\>
```

〔화면 4〕 화면의 하드카피

```
 ALIGN:      (F1)전항목  (F2)다음항목  (F3)선택  (F4)전menu  (ESC H)도움말

단락 align-F5 A    전체 align-Esc F5  Set Justify-F5 J   Set Ragged-F5 R
좌측 지정-Esc L    우측 지정-Esc R    Hard CR-Esc Enter  연속 타자-Esc W
라인 좌측-F5 ←     라인 중간-F5 ↓     라인 우측-F5 →     구역 정렬

라인:  148(총155)  화일명: PRINTER.DOC  모드: [ 테이블 ]  존영역: 9(총391K)
[보통문자][ Ragged align ]
<.:.....+.......+.......+.......+.......+.......+.......+.......+.......+.......
 ........1.........2.........3.........4.........5.........6.........7.........
1234567890123456789012345678901234567890123456789012345678901234567890123456789
   화면에  한글이   나오더라도  프린터는  이  글자가  영문인지  한글인지   전혀
알 수 없다.  어떤  것으로   찍더라도    한글을  찍을려면  프린터  한글  모듈을
한글  상태로  놓거나  프로그램에서  한글  인식코드를   프린터로  보내야   한다.
   위  2  가지  방법  중   아무거나  사용하더라도   한글을    찍는데는    상관이
없으나  어떤  특정한 유틸리티나 그래픽을   사용할  수  없는   경우가   많으므로
이것을  많이  사용하는   사용자라면   프로그램에서   한글인식코드를    프린터로
보내야 한다.^
```

※ 〈보석글 II〉에서 이글을 〈SHIFT〉+〈Print Screen〉을 눌러 하드카피 한 것

hift〉 +〈Print Screen〉 키를 누르면 된다.

하드 카피는 화면의 상태를 그대로 출력하는 것으므로 경우에 따라 아주 유용한 수단이 되기도 한다. 화면 4에 하드카피의 예를 나타내었다.

주기억장치와 보조기억장치

컴퓨터는 본체내에 기억장치를 가지고 있다. 컴퓨터의 카탈로그나 팜플렛을 보면 RAM 640KB, ROM 32KB 처럼 표기되어 있는 것이 있는데 이것이 바로 주기억 장치의 크기이다. 요즈음 보급되는 대다수의 IBM-PC는 640KB의 RAM을 장착하고 있다.(AT 기종에서는 1MB의 RAM을 장착하기도 한다.)

RAM과 ROM은 모두 반도체로 되어있는 기억소자인데 ROM은 컴퓨터를 제작한 회사에서 각종 정보를 기록해 둔 기억소자로서 사용자가 그 내용을 임의로 바꿀 수는 없다.

RAM은 사용자가 관리하는 기억소자로서 거기에 새로운 정보를 수록하거나, 삭제하거나 하는 것은 사용자의 권한이다. RAM과 ROM을 가르켜 주기억 장치 혹은 메모리라고 부르는데, 메모리라는 표현이 더 일반화되어 있다.

이들 메모리 중 ROM은 전원이 없더라도 기억된 내용을 상실하지 않지만 RAM에 수록된 자료는 전원이 끊기면 그 안에 들어있는 모든 자료를 잃게 된다. 그렇기 때문에 RAM에 있는 자료를 영구적으로 보관하기 위해 우리는 보조기억장치라는 것을 이용하게 된다. 컴퓨터의 주기억 공간에 수록된 자료는 전원이 끊기면 모든 내용을 상실하므로 컴퓨터의 사용자는 정전등을 대비하여 수시로 그 내용을 보조기억장치에 속하여 두어야 한다.

컴퓨터의 보조기억장치로는 자기테이프와 자기디스크가 있는데 요즈음은 자기디스크가 보편화되어 있다. 자기디스크는 하드디스크와 플로피디스크로 나누어지는데, 하드디스크는 금속상자에 밀폐되어 있으며 그 내용을 사용자가 들여다 볼 수도 없고, 한번 설치하면 물리적인 변경이 없어도 반영구적으로 사용할 수 있다.

플로피디스크의 경우에는 위에서 설명한 디스켓에 정보를 기록하게 된다. 디스켓은 도너츠형의 얇은 비닐 표면에 산화철가루를 코팅한 것으로, 그 표면을 보호하기 위해 사각형의 자켓으로 포장이 되어 있다. 컴퓨터는 이 원반에 자력을 주입하여 각종 정보를 수록하게 되는데, 컴퓨터의 종류에 따라 디스켓을 취급하는 방법이 다르다. 또 디스켓도 여러 종류가 생산되며 디스켓의 크기에 따라 8인치, 5¼인치, 3.5인치 등으로 구분된다.

또, 한면 만을 사용하느냐 양면 모두를 사용하느냐에 따라 단면과 양면으로 분류되며, 정보를 보통으로 기록하느냐, 2배의 밀도로 기록하느냐, 아주 고밀도로 기록하느냐의 차이에서 단밀도, 배밀도, 고밀도로 분류하게 된다.

디스크의 크기와 기록밀도의 차이는 자기가 사용하고 있는 컴퓨터의 드라이브 성능에 따라 결정되는 만큼 디스크를 구입하려는 경우에는 이런 사항을 충분히 알고 구입하여야 한다. 대부분의 PC에서는 5¼인치 양면 배밀도 디스켓을 사용하며, 디스켓의 포장지에 2S 2D 라고 표기되어 있다. 요즈음은 배밀도의 디스켓을 사용하는 PC가 보급되고 있으므로 이 경우에는 2S HD를 선택하도록 한다.

디스켓의 초기화

생산 공장에서 처음 출하된 디스켓은 컴퓨터가 자료를 수록할 수도, 읽어낼 수도 없는 이른바 정보의 진공 상태로 되어 있다. 이 디스켓은 수록하는 방법에 따라 더 많은 자료를 수록할 수도 있고, 더 적은 자료를 수록할 수도 있다. 컴퓨터는 자신이 사용하는 디스켓이 어떤 상태로 자료가 수록되어 있는지를 알고 있어야만 그 디스켓에 수록된 자료를 제대로 읽어들일 수 있게 된다. 따라서 모든 디스켓은 컴퓨터에서 필요한 최소한의 정보를 기록해 둘 필요가 있다. 이 과정을 포매팅이라고 하며, 디스켓은 일단 포매팅이라는 과정을 거쳐야만 비로소 자료를 수록할 수 있게 된다.

디스켓을 포맷시키면 컴퓨터는 디스켓을 일정한 크기의 영역으로 나누고, 각각의 영역에 차례대로 번지를 붙이게 된다. 이 일련의 번지들을 섹터라고 부른는데 2S SD인 경우 0~719의 섹터가 할당되게 된다. 만약 자료가 수록되어 있는 디스켓을 포맷시키면 거기에 수록되어 있는 모든 자료를 잃게 되므로 주의하여야 한다.

디스크를 포맷하는 과정은 DOS디스크를 A:에 넣고 A〉FORMAT B:를 입력한다. 컴퓨터는 B:에 새 디스켓을 넣고 키를 치라는 메시지를 출력하는데 이때 B:에 새 디스켓을 넣고 키를 치면 된다. 플로피디스크 드라이브가 1개 뿐인 기종에서는 B:에 새 디스켓을 넣고 키를 치라는 메시지가 출력되면 A :에 있는 DOS디스켓을 제거하고 새 디스켓을 넣은 후에 키를 치면 된다. 이때는 드라이브가 B:로 지정되게 된다.

드라이브가 1개 뿐인 기종에서도 드라이브가 2개인 것 처럼 사용하여야 한다. 그러면 컴퓨터가 일아서 A:, B:로 구분하여 사용하며 그 때 마다 디스켓을 넣으라는 메시지를 출력하므로 적당히 디스켓을 교환하여 주면 된다.

디스켓의 포맷은 현재 드라이브에 들어있는 DOS디스크의 디렉토리에 FORMAT. COM이라는 화일이 존재하여야만 가능하므로, 자신이 사용하는 DOS디스켓이 2장으로 구성된 경우에는 어디에 FORMAT.COM이 들어있는지 확인하도록 하라.

포맷이 끝나면 컴퓨터는 다른 디스켓을 포맷하겠느냐고 물어오는데 이때

〔화면 5〕 디스크의 포매팅

```
A>FORMAT B:                                  ← 입력
Insert new diskette for drive B:   ┐
and strike ENTER when ready        ┘
                                             ↵ 을 누른다.
Head:   0 Cylinder:   3
                                             Format과정 표시
Format complete

   362496 bytes total disk space  ┐
   362496 bytes avaiable on disk  ┘── Format완료 메시지

Format another (Y/N)?                        ← 'N'을 눌러 DOS로 간다.
```

Y나 N을 입력하도록 한다.

화면 5에 포맷하는 과정을 표시했으니 참고하기 바란다.

화일의 복사

디스켓에 수록된 자료는 만일을 대비하여 복사본을 만들어 두어야 한다. 디스켓은 물리적이나 자기적인 손상을 입게된면 그 안에 수록된 자료는 영원히 회복할 수가 없게 된다. 이것을 대비하는 것은 전적으로 사용자의 책임이다.

화일을 복사할 경우에는 COPY 명령을 이용한다.

COPY명령의 형식은 다음과 같다.

A>COPY [드라이브명:]화일명.확장자 [드라이브명:][새이름]

여기서 [ ]안에 내용은 생략이 가능하다는 의미이다. 생략이 가능하기는 하지만 셋을 모두 생략하면 안된다. COPY 명령은 지정한 화일을 다음에 지정한 드라이브에 새로운 이름으로 복사하라는 명령이다. 새로운 이름을 지정하지 않으며 먼저의 이름으로 복사된다. 와일드카드 문자를 써서 여러개의 화일을 복사하는 경우에는 DOS는 디스크에 먼저 수록된 화일부터 차례대로 복사를 하게 된다.

COPY 명령은 다음과 같은 방법으로 쓰이게 된다. (화면 6 참고).

COPY 명령은 디스켓에 수록되어 있는 자료를 화일단위로 복사한다. 그러므로 필요한 화일만을 복사할 수 있다. 그러나 화일에 관계없이 디스켓을 송두리째 복사하여 원래의 디스켓과 똑같은 디스켓을 만들고자 할 경우에는 DISKCOPY 명령을 이용한다. A〉DISKCOPY A: B: 를 입력하면 컴퓨터는 A:에 디스켓의 원본을, 그리고 B:에 디스켓을 넣어달라는 메시지를 출력하게 된다. 이때 각각의 드라이브에 디스켓을 넣고 키를 치면 A디스켓과 똑같은 내용이 B디스켓에 복사된다.

만약 드라이브가 1대 뿐일 경우에는 A:드라이브에 원본을 넣고 키를 친다. 그러면 잠시후에 B:에 새 디스켓을 넣어달라는 메시지가 출력되므로 이때 A:의 원본을 빼내고 새 디스켓을 넣고 ↵ 키를 입력하면 된다.

DISKCOPY 명령을 수행하기 위해서는 FORMAT의 경우와 같이 현재의 DOS 디스크 상에 DISKCOPY.COM 이라는 화일이 존재 하여야 하므로 먼저 자신의 DOS 디스크에 어떤 화일들이 수록되어 있는지를 확인한다.

컴퓨터로 기안용지를 작성한다거나 공문서를 만든다면 그 자료들은 어떤 내용이 되든지 디스켓에 화일이라는 형태로 수록할 것이다. 이 자료들은 경우에 따라서는 앞으로 전혀 쓸모 없는 화일이 되어 괜히 디스크의 기억공간만 차지하게 된다. 이러한 화일들은 디스크에서 삭제를 하여야 한다.

화일의 삭제

화일의 삭제는 DOS의 DEL(Delete) 명령이나 ERASE 명령으로 가능하다.

```
A>DEL COMMAND.COM
A>DEL B:*.EXE
B>DEL A:KIM.*
B>DEL *.*
```

와 같은 형식으로 쓰인다.

화일은 한번 삭제하면 DOS에서는 복구할 수 없으므로 신중히 생각한 후 삭제하여야 한다.

DEL *.*처럼 모든 화일을 삭제하려는 경우에 컴퓨터는 'Are you sure (Y/N)'과 같은 내용을 출력하는데 'Y'를 입력하면 모든 화일은 지워진다.

화일이름의 변경

DOS는 사용자가 필요에 의해 화일

〔화면 6〕

A>COPY COMMAND.COM B:	(A:의 COMMAND.COM을 동일한 이름으로 B:에 복사한다.)
A>COPY *.COM B:	(A:의 확장자가 COM인 화일은 모두 B:에 같은 이름으로 복사한다.)
A>COPY *.DAT B:	(A:의 확장자가 DAT인 화일은 모두 B:로 복사한다.)
A>COPY B:KIM.DAT A:KIM-1.DAT	(B:에 있는 KIM.DAT를 A:에 KIM-1.DAT라는 이름으로 복사한다.)
A>COPY *.* B:	(A:의 모든 화일을 B:로 복사한다.)

〔표 1〕 에러 메시지

에 러 메 세 지	의 미	조 치
Bad command or file name	명령이나 화일명이 잘못 지정되었다.	철자를 확인하여 다시 입력한다.
File not found	디스켓에 화일이 하나도 없다.	
Duplicate file name or file not found	RENAME으로 지정한 화일이 없거나, 새로운 이름과 동일한 화일이 이미 존재한다.	화일 이름을 확인한 후 다시 입력한다.
Not ready error reading drive A : Abort, Retry, Ignore	드라이브의 도어가 열려 있다.	도어를 닫고 'R'을 입력한다.
General Failure error reading drive A: Abort, Retry, Ignore	디스크가 포매팅이 되지 않았거나 디스크에 물리적인 흠이 있다.	'A'를 입력한 후 디스켓을 포매팅 시킨다.
Write fault error writing device PRN Abort, Retry, Ignore	프린터가 켜지지 않았거나 'Ready' 상태가 아니다.	프린터를 점검하고 'R'을 입력한다.
Invalid number of parameters	파라메터의 지정이 잘못되었다.	드라이브나 화일이름을 정확히 입력한다.

이름을 변경하는 것을 허용한다. 이름의 변경은 REN(Rename) 명령에 의해 가능하다. 'A〉REN COMMAND.COM KIM.DAT '에 의해 COMMAND. COM이 KIM.DAT로 이름이 바뀌게 된다. 즉 앞의 이름을 뒤의 이름으로 바꾸게 되는 것이다. 이때 지정한 새이름과 같은 이름의 화일이 디스켓에 있다면 컴퓨터는 에러 메시지를 출력하고 REN 명령을 수행하지 않게 된다.

REN 명령은 와일드카드 문자를 써서 여러개의 화일이름을 한 번에 바꾸는 것을 허용한다. A〉REN KIM-??.DOC *.DAT 에 의해 KIM으로 시작하는 화일의 확장자가 DOC에서 DAT로 바뀌게 된다.

프로그램의 실행

위에서 확장자명이 COM, EXE, BAT인 화일은 DOS에서 실행가능하다는 것을 배운 바 있다. 이런 프로그램은 DOS에서 화일명을 입력하는 것만으로 곧바로 실행하게 된다. DOS디스켓에는 대부분 베이직을 실행할 수 있는 프로그램이 수록되어 있다. GWBASIC.EXE이 바로 그것인데 BASICA.COM, BASIC.COM 등으로 수록 되어있는 경우도 있다. 이들은 확장자명이 EXE이거나 COM이므로 A〉GWBASIC 으로 실행하게 된다.

이제는 DOS를 떠나서 베이직 상태에 와있으므로 베이직을 배운 독자라면 베이직 프로그램을 작성하거나 실행할 수 있게 된다. 베이직에서 DOS로 복귀하려면 SYSTEM을 입력하면 된다.

만약 화일이름이 같고 확장자명이 다른 화일이 디스켓에 있다면 COM, EXE, BAT 순으로 우선 순위가 결정되어 있다. 즉, ABC.COM과 ABC.EXE가 같은 디스켓에 있다면 ABC를 입력하면 COM이 EXE보다 우선 순위가 높으므로 ABC.COM이 실행되게 된다. 다시 말하면 ABC.EXE는 실행시킬 수 없다는 말이다.

마찬가지로 ABC.EXE와 ABC.BAT가 디스켓에 있다면 ABC.BAT는 실행시킬 수 없다는 결론이 나온다.

이것은 배치화일의 경우에도 적용이 되므로 DOS에서 실행하는 화일은 그 확장자에 의해서 우선 순위가 결정되어 있다는 사실을 유념하여야 한다.

에러 메시지

DOS는 다양한 에러 메시지를 출력한다. 한글 DOS의 경우에는 대부분의 메시지가 한글로 나오기 때문에 별로 어려울 것은 없으나 영문 DOS의 경우에는 무슨 의미인지를 몰라 아주 당혹스런 경우가 왕왕있다. 지금까지 배운 명령어를 실행하는 과정에서 발생할 수 있는 에러메시지와 그 의미를 표 1에 정리해 두었으니 참고하기 바란다. 다음 호에는 좀 더 재미있고 유익한 명령들을 소개 할 예정이다. PC

〈용어해설〉

● 액세스 타임(Access Time)

호출시간이라고도 한다. 메모리나 주변기에 대해서 데이터의 번지를 지정한 다음 그 데이터가 실제로 얻어지기까지의 시간 또는 전송이 개시되기까지에 요하는 시간. 이 시간이 짧으면 그만큼 전자계산기의 고속화에 도움이 되므로 성능을 결정하는데 중요한 조건으로 되고 있다. 사이클 타임은 1사이클의 조작이 완전히 끝나기까지 요하는 시간으로 메모리의 성능에 대해서는 액세스 타임/사이클 타임의 양자를 병기하는 경우가 많다.

● 부트스트랩(Bootstrap)

프로그램을 입력하는 방법의 하나로서 최초에 명령을 리드하기 위한 간단한 조작을 해두면 그 다음부터는 그 명령의 리드를 순차적으로 행하여 최종적으로는 완전한 프로그램이 기억장치 내에 수용되도록 만들어진 루틴.

둘리가 본 컴퓨터세계 1

컴퓨터 길들이기

이만희/한국교육개발원 컴퓨터교육 연구센터

컴퓨터를 친구로 하기 위하여

TV 안테나선조차 제대로 연결 못하고 전기스위치만 여러개 달려 있어도 기가 죽어버리는 대부분의 사람들은 컴퓨터 앞에 서면 전생에 무슨 큰 잘못이라도 진 것처럼 슬슬 피하는 경우가 많다. 더구나 불행하게도 컴퓨터가 우리 앞에 나타나 그 위력을 과시하는 경우가 점점 많아져 가뜩이나 주눅들게 한다.

컴퓨터! 그는 이 세상에 나타난 지 얼마되지 않으면서도 너무나 빨리 사람들 사이에 끼어들고 있다. 특히 몇몇 친한 사람들에게는 마치 성실한 하인처럼 시원 시원하게 일을 해주지만 컴퓨터를 어려워하는 순진하고 얌전한 사람들이 만지기만 하면 영낙없이 Error! 하면서 무섭게 으르렁거리는 사나운 세퍼트 같아서 쉽게 접근하기 어렵다.

간혹 주위에서 컴퓨터를 자기 수족처럼 부리는 사람들이 있다. 그들은 어쩌면 인류가 말을 보았을 때 용기를 내어 최초로 용감하게 말위에 올라탄 사람들과 같을 것이다.

현대는 정보의 홍수 또는 "정보화 시대"라고 한다. 요즘은 꼬마 아이도 그동안 발전된 또 새롭게 추가된 정보를 배우기 위하여 정신없이 유치원을 가고 속셈학원을 다니고 모두 다 바쁘게 돌아가고 있다. "아는 것이 힘이다"라는 말은 정보를 보다 빨리, 보다 많이 아는 사람이 이 세상을 힘있게 살아가고, 정보에 어두우면 약자의 서러움을 겪게 된다는 뜻이 된다. 우리 주위에서 흔히 보는 퍼스널 컴퓨터에 잠재된 그 힘도 정보의 밀림에서는 호랑이의 그것보다도 훨씬 강력하다.

특히 새로운 지식이 넓은 신대륙처럼 펼쳐지는 오늘날은 정보의 숲과 지식의 들판을 말처럼 타고 달릴 수 있는 새로운 친구, 바로 이 "컴퓨터"의 도움이 반드시 필요하다. 그가 없이는 길을 잃어버리고 야수의 밥이 되어 버릴 수 밖에 없다. 그 불쌍한 먹이를 찾아 나타나는 사나운 늑대 또한 컴퓨터일 것이다.

컴퓨터, 그는 길들여져야만 우리의 친구가 될 수 있는 살아있는 기계이다. 컴퓨터가 살아 있다는 말을 들을 수 있는 것은 우리와 대화를 주고 받기 때문이다. 컴퓨터를 친구로 하기 위해서는 컴퓨터의 몸체 구조를 알고 생각을 이해해야 하고 계속 관심을 갖고 대화해야 한다. 관심을 갖고 용기를 내어 손을 내민다면 처음은 서먹 서먹하지만 결국은 컴퓨터의 말문이 열릴 것이다.

어쩌면 우리의 접촉을 기대하며 생명을 넣어 주기를 기다리고 있는지도 모른다.

정보의 숲 속에서 노려보고 있는 사나운 야수의 눈초리도 사랑으로 길들이면 집에서 기르는 강아지로 변하듯이 컴퓨터를 이해하고 사랑한다면 그는 우리를 도와주며 날개 달린 유니콘처럼 우리를 태우고 새로운 컴퓨터피아로 안내할 것이다.

컴퓨터의 신체구조

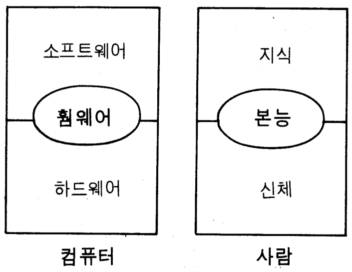
컴퓨터를 길들이기 위해서는 먼저 컴퓨터의 신체구조 및 사고방식을 이해해야 한다. 컴퓨터가 다른 기계와 달리 길들여 진다는 것은 몸과 마음 즉, 하드웨어와 소프트웨어로 이루어져 있기 때문이다. 또 훰웨어라고 고정된 소프트웨어도 있다. 훰웨어는 신체의 반사작용이나 본능과 같이 컴퓨터를 만들때 아주 하드웨어의 일부분으로 고정시킨 소프트웨어인데 컴퓨터에 전기가 들어가면 자동적으로 컴퓨터 자신의 상태를 점검하고 명령을 기다리도록 하는 부분들과 같이 컴퓨터가 기본적으로 살아 움직이기 위해 반드시 있어야 하는 부분이다(그림 1 참조).

컴퓨터는 사람의 능력인 기억, 계산, 처리기능을 비슷하게 흉내내는 전자장치이기 때문에 내부 구성은 사람의 두뇌가 갖고 있는 기능을 수행하도록 비슷한 구조로 되어 있다.

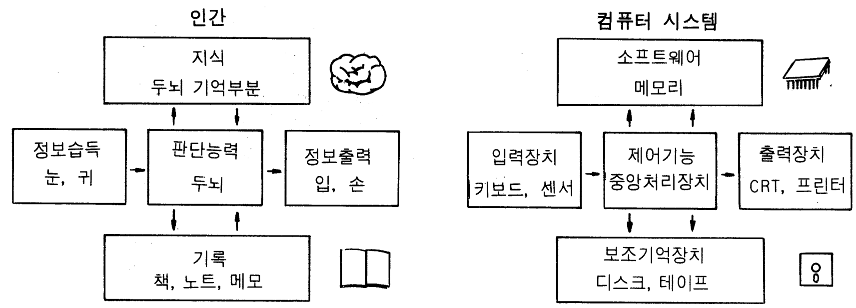
먼저 간단하게 컴퓨터의 구조를 살펴보면 CPU(중앙처리장치), 메모리(기억장치), 입력장치, 출력장치 그리고 보조기억장치 다섯가지로 나눌 수 있다. 사람이 정보처리를 두뇌로 하듯이 컴퓨터는 중앙처리장치(CPU)로

"이세상에서 가장 슬픈 컴퓨터는 관심받지 못하는 컴퓨터이다."

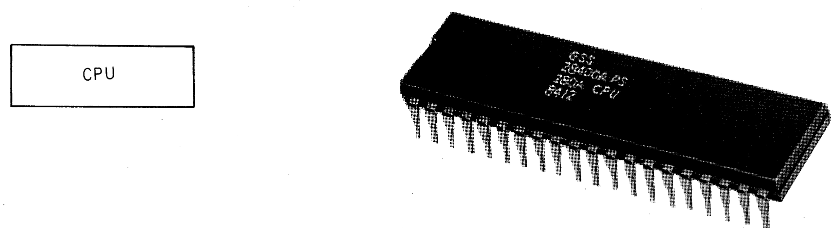
〔그림 1〕



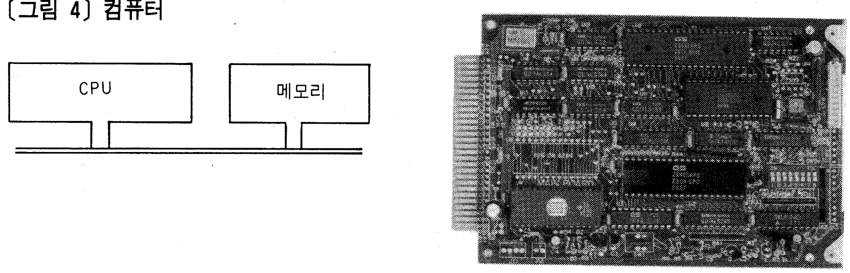
〔그림 2〕 인간의 정보처리 시스템과 컴퓨터 시스템



〔그림 3〕 마이크로 프로세서



〔그림 4〕 컴퓨터



한다.

잠시 생각하는 것은 머리속에 담아두다가 잊지않기 위해 장부나 노트에 기록하듯이 컴퓨터도 내부 메모리와 보조 기억장치를 이용해서 정보를 보관한다. 사람은 처리된 정보를 음성이나 표정 또는 손으로 표현하는데 비해 컴퓨터는 화면장치나 프린터, 플로피디스크 등으로 출력한다. 이와 같이 사람과 컴퓨터는 유사한 처리 장치를 갖고 있다(그림 2 참조).

의미를 명확히 하기위해 퍼스널 컴퓨터를 예를 들어 다시 구성을 하면 다음과 같다.

첫단계는 마이크로 프로세서(Micro Processor) 단계이다. CPU 즉, 중앙처리장치만 있는 것으로 이것만 가지고는 컴퓨터라 할 수 없다. 마이크로 프로세서는 보통 하나의 IC칩으로 되어 있는데 내부에는 수만~수십만의 트랜지스터가 모여 있다.

요즘은 이 마이크로 프로세서가 수많은 곳에서 부품으로 사용되고 있다. 냉장고, 세탁기, 자동차까지 조금 복잡한 기계에서는 대부분 마이크로 프로세서를 사용해 성능을 크게 향상시키고 있다(그림 3 참조).

둘째는 컴퓨터 단계이다.

메모리를 마이크로 프로세서에 연결시키면 가장 기본적인 컴퓨터라 할 수 있다. 이때 프로세서는 판단 처리기능, 메모리는 기억기능을 수행하지만 이것은 외부로부터 입력을 받거나 결과를 나타낼 수 없기 때문에 실제로 독자적으로 일을 처리할 수는 없다(그림 4 참조).

세째는 컴퓨터 시스템 단계이다.

마이크로 프로세서와 메모리에 입력, 출력장치가 추가된 것으로 이 단계야 말로 비로서 컴퓨터로서의 역할을 제대로 할 수 있다. 우리가 흔히 컴퓨터 혹은 퍼스널 컴퓨터라고 부르는 것은 정확히 말하면 이 컴퓨터 시스템을 가르키는 것이다. 여기에서 메모리에 적당한 소프트웨어가 채워지면 완전한 컴퓨터 시스템이 되어 입력을 받아 처리해서 다시 출력을 보내는 살아있는 멋진 기계가 되는 것이다(그림 5 참조).

마지막으로 원칩컴퓨터(One Chip Computer)라는 것도 있다.

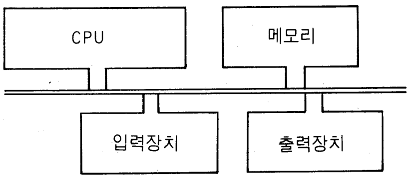
말 그대로 하나의 IC안에 CPU, 메모리, 입출력 회로를 전부 모아 놓은 것이다. 기억용량도 작고 능력이 떨어지지만 하나의 IC로 컴퓨터의 역할을 다 할 수 있으므로 다양한 곳에 쓰일 수 있다(그림 6 참조).

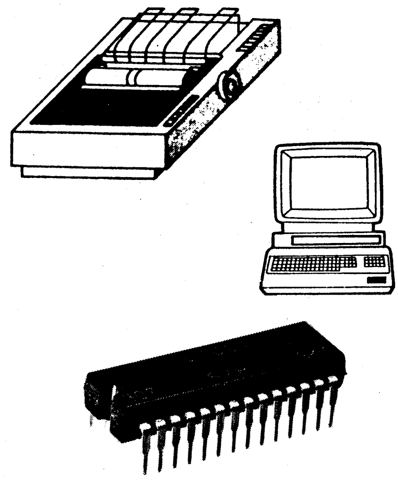
컴퓨터 해부학

고등학교 생물시간에 동물의 해부도를 공부한 적이 있을 것이다. 그것을 가지고 외과 수술을 한 것은 아니었지만 알아두어 오래 살다보면 결국은 도움을 받았을 것이다.

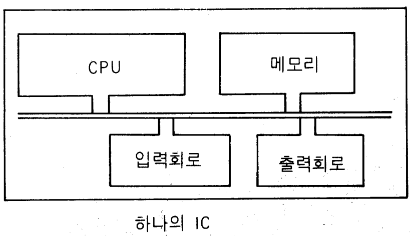
컴퓨터에서 가장 특징이 되는 메모리와 가장 복잡한 CPU를 조금만 해부해 보겠다. 생물 점수가 좋지 않았어도 무사히 고등학교를 졸업할 수 있었듯이 완전히 이해하지 못해도 이 코스는 마칠수 있으니 안심하고 침착하게 읽

〔그림 5〕 컴퓨터 시스템

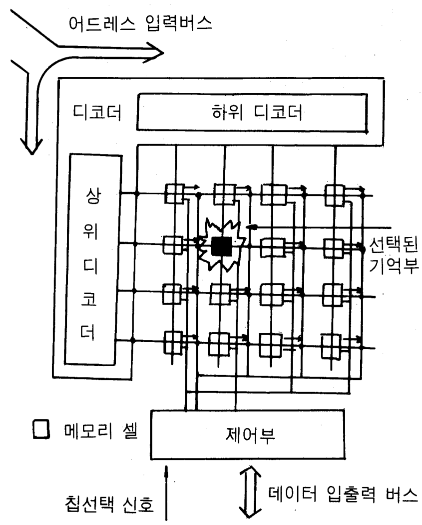




〔그림 6〕 원칩 시스템



〔그림 7〕 메모리의 구성



어주기 바란다.

(1) 메모리의 해부

우리의 머리속은 생각으로 가득 차 있다. 형제나 친구의 얼굴도 기억을 하고 있기 때문에 알아볼 수 있고 말도 글자도 머리속에 담겨 있기에 사용할 수 있다. 컴퓨터는 어떻게 기억하는 것인가?

컴퓨터의 기억장치의 원리는 바둑판과 같다. 바둑판에는 일정하게 가로 세로로 줄을 그려 위치를 구별하고 그 위에 백돌과 흑돌이 놓여진다. 하나의 지점 위에는 하나의 돌만을 올려 놓을 수 있다. 백돌을 놓아야 되는데 만일 먼저 흑돌이 놓여 있으면 그 돌을 치워야만 백돌을 놓을 수 있다. 이렇게 한점뒤에 백돌이나 흑돌을 하나씩 올려 놓아 그것의 조합으로 기억을 하는 것이다.

스무고개 놀이를 할 때 예, 아니오만으로 무엇인가를 알아 맞추어가는 것처럼 실제로는 예, 아니오를 적당히 조합하면 어떠한 것이라도 표시할 수 있다.

컴퓨터에서는 바로 이 두가지의 상태 즉 예, 아니오를 대신한 1과 0을 가지고 이세상의 모든 것을 처리한다.

그림 7은 메모리의 구성을 나타낸다.

●기억부(Cell)

바둑판에 있는 하나의 점과 같이 하나의 정보 즉 0이나 1인가를 표시한다. 전자회로중 플리플립이라는 것이 있는데 이것은 바둑판위에 하나의 정보를 표시하는 스위치와 같은 역할을 한다. 즉 전압이 올라가거나 (+5V) 내려가거나 (0V)한다. 둘중에 한가지만 가능하고 중간은 있을 수 없다.

초기의 컴퓨터는 릴레이라고 하는 전기스위치의 조합으로 구성되었는데 벌레가 이 스위치 틈으로 자주 끼어 들어가 고장을 일으켰다.

이 때문에 디버그 즉, 벌레 잡아내기라는 말이 생겼다.

●번지지정부(Address Decoder)

하나의 메모리 IC안에 수많은 메모리의 셀들이 바둑판처럼 가지런히 줄을 맞추어 놓여 있다. 그중에 하나의 기억장소를 찾아가기 위해서 번지수를 지정해야 한다. 어드레스 디코더에 의해 주소가 지정되면 그 위치의 기억부가 신호를 내보낼 수 있게 된다.

●제어부

메모리의 한부분이 지정되면 그 안에 들어있는 내용을 읽어 낼 것인지 새로 내용을 바꾸어 놓을 것이지(READ, WRITE) 제어신호를 보낸다. 물론 이 신호는 CPU에서 받아온다. 이 제어부의 선은 전체 기억부에 모두 연결되어 있지만 어드레스 디코더에 의해 지정된 부분만 동작을 한다.

(2) CPU(마이크로 프로세서)의 해부

마이크로 프로세서는 간단한 것도 만 오천에서 30만개 정도의 트랜지스터 조합으로 구성되어 있다. CPU(Central Processing Unit)의 내부를 살펴보면 기본적으로 **그림 8**과 같은 구조를 이루고 있는데 그 역할은 다음과 같다.

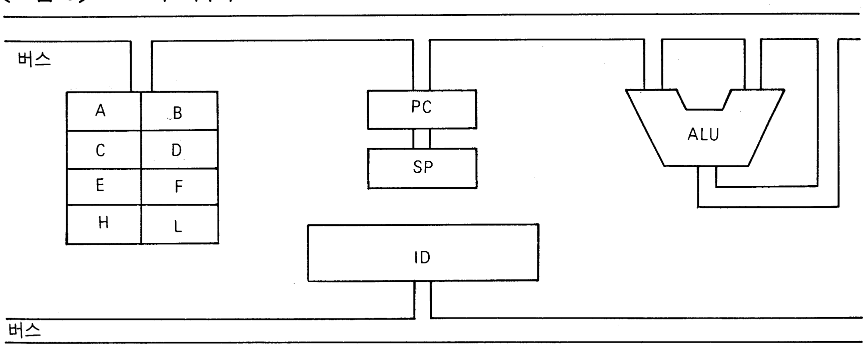
●어큐뮬레이터 및 범용 레지스터

CPU내에서도 가장 바쁜 곳으로 사람에 비유하면 흘러가는 생각이 잠시 머무르는 곳이다. 컴퓨터가 수를 계산할 때도 명령을 일단 이곳에 놓아야만 그 의미를 해석하고 다음에 숫자를 옮겨 놓아 계산을 하게 된다. 어큐물레이터(Accumulator)는 이를 보조하는 여러 범용 레지스터(Register : A,B,C, D,E,F,H,L)들과 함께 작동하여 능률을 높이기도 한다(**그림 9** 참조).

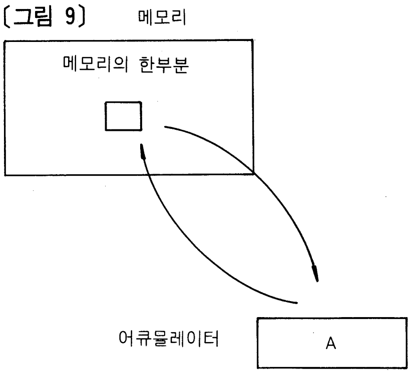
●PC(Program Countr)

다음으로 수행할 지시가 담긴 기억장소를 나타내고 있는 곳이다. 하나의 명령이 실행되면 자동적으로 숫자가 증가되어 다음의 장소를 가리킨다. 이 곳에 자동으로 증가하는 숫자가 아닌

〔그림 8〕 CPU의 내부구조



〔그림 9〕



호출하고 싶은 메모리의 장소를 가르키는 숫자를 넣으면 프로그램은 다른 기억장소로 건너 뛰어가서 수행하게 된다.

● SP

Stack은 쌓아 놓는다(퇴적)는 뜻이다. 사람이 책을 읽다가 전화를 받아야 할 필 요가 있으면 읽던 책을 옆에 잠시 놓아 두었다가 전화를 먼저 받고 마치고 돌아와서 다시 일을 시작하듯이 잠시 밀어둔 일의 주소를 가리키고 있는 곳이다. 이때 몇가지 일들이 겹쳐서 밀리게 되면 밀리는 순서대로 주소를 차례차례 쌓아 놓아 다시 순서대로 돌아가게 된다.

사람의 경우 몇가지 일이 계속 발생하면 처음에 하던 일을 잊어 버리고 마는 수가 있지만 CPU는 이 SP(Stack Pointer)의 도움으로 차례 차례 다시 돌아오게 된다.

● ALU

여러개의 계산 회로도들로 구성되어 있다. 컴퓨터는 전자계산기에서 발전한 것인데 ALU(Arithmetic Logic Unit)는 CPU가 내부에 지니고 있는 계산기라고 할 수 있다.

컴퓨터의 계산은 기본적으로 덧셈과 뺄셈을 이용하여 곱하기, 나누기등을 한다. 그림 8에서 ALU의 표시는 두개의 숫자가 들어가서 하나의 숫자로 나온다는 뜻이다.

● ID

컴퓨터를 이야기할 때 명령어 또는 명령어를 수행한다는 말이 자주 나온다. 명령을 수행하기 위해서는 명령의 의미에 따라 각 부분이 손발을 맞추어 움직여야 하는데, 숫자로 표기된 명령이(ex : 111011101 : LDA, I) 어큐물레이터에 들어가면 ID(Instruction Decoder) 에서 그 의미를 해석한다. 즉, 어느 명령에 대해서는 CPU의 각 부분이 어떻게 움직인다는 신호를 만들어 내보내는 곳이다.

● 버스회선(Bus Line)

컴퓨터 내부에서 여러개의 부분들이 서로 신호와 데이터를 교환하기 위해서는 통신선로를 필요로 한다. 그러나 필요한 선로를 각각의 부품들이 서로 연결하기 위해서는 많은 공간을 필요로 한다. 그래서 공동으로 사용하는 복도와 같이 하나의 선로다발을 묶어서 연결하고 순서를 정하여 그 공동선로를 이용한다. 이러한 공동선로를 버스라 부른다. 이 버스 기술은 수많은 부분 및 부품들을 효율적으로 관리하는 방법이다. 최근에 나오는 컴퓨터 내부에는 슬롯이라고 부르는 다른 장치를 연결하는 부품이 있는데 이도 마찬가지로 버스에 연결되도록 되어 있다.

이밖에도 CPU에는 인덱스 레지스터(Index Register), 플래그 레지스터(Flag Register)등 여러 부분이 있다. 그러나 동작원리는 이제까지 설명한 내용과 큰 차이가 없다. 다만 능률(속도)을 향상시키기 위하여 많은 보조 부분이 추가되어 있다(그림 10 참조).

컴퓨터가 똑똑한 이유 (CPU 동작원리)

흔히 컴퓨터에 대해서 그 특징을 말할때 일초에 수십만회 혹은 수억회까지의 빠른 속도로 계산을 할 수 있다는 놀라운 신속성과 함께 지치지 않고 수많은 숫자나 정보를 처리하는 정확성을 든다. 그러나 이러한 신속성과 정확성은 이제까지 인간이 발명한 여러가지 기계들도 갖고 있었다. 새롭게 등장한 컴퓨터에는 이 두 가지와 더불어 또 다른 큰 특징이 있다. 바로 상황 대처 능력이다.

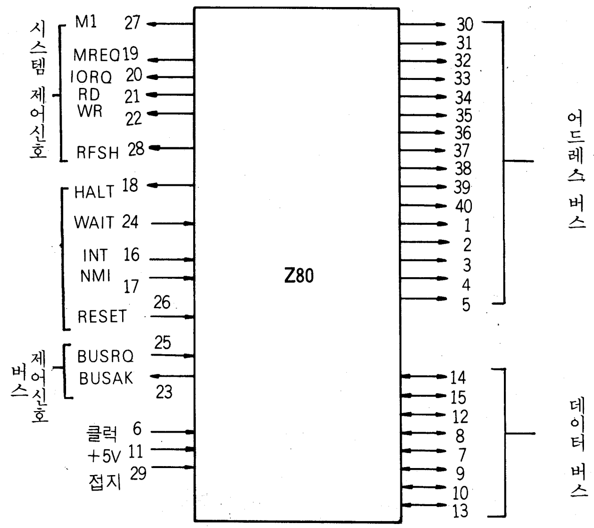
사람은 한 가지 일을 하고 있는 중에 전화가 오면, 하던 일을 잠시 멈추고 전화를 받은 다음에 다시 일을 계속한다. 즉 상황에 따라 적절하게 행동을 변화시킨다. 이러한 행동을 우리는 융통성이 있다고 한다.

컴퓨터도 이와 유사하게 일을 처리하는 중에도 상황에 따라서 진행하던 일을 잠시 밀어두고 보관(기억)되어 있는 프로그램으로 상황에 적합한 처리를 한 후에 다시 하던 일로 돌아와 계속할 수 있다.

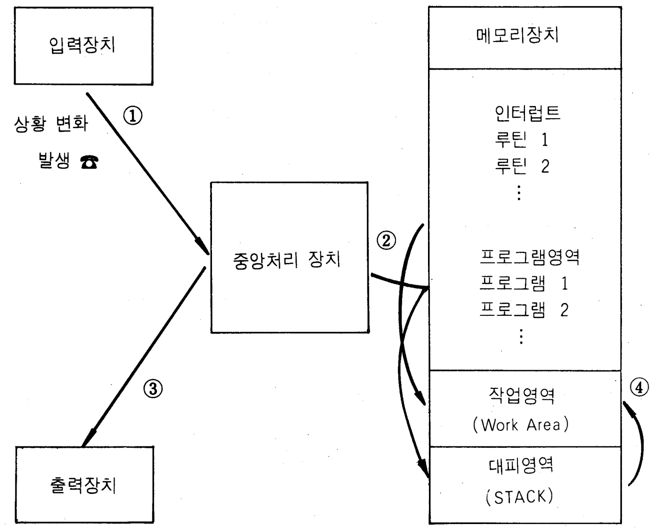
이렇게, 다른 기계에서는 볼 수 없는 신기한 일이 가능한 것은 생각하는 부분, 즉, 중앙처리장치(CPU)가 있기

"사람이 그 형상을 본따 컴퓨터를 만드니 그 모습이 보기에 신기하도다."

〔그림 10〕 Z80의 CPU



〔그림 11〕 상황 변화에 따른 대처과정



때문이다.

여기서는 이 CPU의 내부 구조 및 컴퓨터의 동작원리를 요즘 널리 보급되고 있는 소형 컴퓨터의 CPU를 중심으로 간략하게 설명하고자 한다.

상황 변화에 대처하는 과정을 통해서 각 장치의 역할을 살펴보면 다음과 같다(그림 11 참조).

① 한가지 작업의 수행중, 입력장치가 상황의 변화를 감지하면 인터럽트(Interrupt : 가로채기 신호)를 CPU로 전송한다.
② CPU가 인터럽트 신호를 받으면, 수행중인 작업을 메모리의 특정 부분(스택)에 기억시키고, 변화상황에 대비해서 준비해 놓은 적합한 프로그램(인터럽트 루틴)을 메모리에서 읽어 온다.
③ 메모리에서 읽어온 프로그램을 수행한 후 CPU는 필요하면 출력장치로 처리된 반응을 내보내고, 필요한 경우, 상황 대처과정에서 발생한 새로운 정보를 메모리에 저장한다.
④ CPU는 메모리의 스택 부분에 기억시켰던 작업을 다시 찾아와서 하던 일을 계속한다.

이와 같은 일련의 과정이 상당히 짧은 시간(몇백분의 일초)에 일어나기 때문에, 여러가지 변화가 동시에 일어나더라도 각각 적합한 상황 대처를 하면서 아무 일도 없었던 것처럼 다시 일을 계속할 수 있는 것이다. 이때 반드시 상황에 대비하여 미리 준비된 프로그램이 있어야 한다. 만일 준비해 놓지 않는 상황이 발생하는 경우는 다른 기계보다도 더욱 멍청한 기계가 되고 만다.

프로그램을 잘 이용하면 컴퓨터가 상황을 처리하는 과정에서 발생한 새로운 정보를 메모리에 추가로 기억시켜 다음 상황에서는 더욱 세련되게 반응할 수 있도록 하기도 한다.

즉, 컴퓨터는 '학습을 할 수 있는 기계'라고 말할 수 있다. 컴퓨터의 능력을 스피드와 용량만으로 말할 수 없는 이유가 바로 그것이다. 컴퓨터는, 기본소질(스피드, 용량 : 하드웨어)과 이제까지 경험(소프트웨어)이 잘 조화되어 있을 때 그 능력을 훌륭히 발휘할 수 있다.

처음 컴퓨터가 탄생했을 때는 거의 하나에서 열까지 일일이 프로그램을 컴퓨터에 매번 입력해야 했으나 그동안 자주 일어나는 상황은 이에 대비하여 프로그램을 모아 놓기 시작했다. 이를 O/S(오퍼레이팅 시스템)라 부른다. 따라서 O/S가 발전하면서 컴퓨터는 점점 세련되어 가고 있다.

많이 알고 있는 사람이 강력한 힘을 발휘하듯이 컴퓨터도 점점 똑똑해져 잘 길들여지고 몇배나 강력한 컴퓨터로 변하고 있는 것이다.

그러나 컴퓨터만 사람에 맞추어 길들여 지기를 바란다면 무리일 것이다. 동물을 아무리 길들여도 야수의 특성이 남아 있듯이 컴퓨터도 기계적인 특성을 버릴 수는 없다. 차라리 사람이 그 기계적인 특성을 이해하고 적당히 컴퓨터의 기분을 맞추어 주는 편이 편하다. 컴퓨터를 길들인다는 것은 우리 스스로도 컴퓨터에 길들어져야 한다는 말도 된다. PC

〈참고문헌〉

1. Z-80 마이크로 컴퓨터, 이태원역, 가남사
2. 마이크로 컴퓨터 시스템 입문, 정태충, 마이크로 소프트웨어
3. 정보기술 동향과 우리의 대응책, 한국전자통신 연구소

어떻게 시작하나요 ③

화일관리

이번호에는 DOS의 명령어를 이용하여 자신의 화일들을 효율적으로 관리하는 방법에 대해서 알아 보자.

화일을 관리하는 DOS의 기본 명령어에는 COPY, COMP, RENAME, ERASE, DELETE, 그리고 VERIFY 등이 있으며 보통은 화일이름을 가지고 작업을 행하게 된다.

화일에 관련된 DOS명령어들을 사용하기 전에 기본적으로 화일이름에 관한 규칙을 알아야 하는데 그 형식을 보면 다음과 같다.

[드라이브명:][패스][화일이름][.확장자명]

① 해당 화일들이 들어 있는 디스크드라이브의 드라이브 명(A:, B: 등)
② 현재 디렉토리에서 해당 화일이 들어 있는 부디렉토리까지의 경로(패스명)
③ 화일이름은 1~8자 이내로 만든다.
④ 화일 확장자명은 화일이름 뒤에 점을 찍고 1~3문자로 지정한다.(.COM, .EXC 등)

예) A:\BUDGET88.WKS

화일 작업시, 같은 유형의 화일들은 와일드카드 문자(*,?)를 사용하여 쉽게 지정할 수 있다. 쉬운 예로 현재 작업중인 드라이브의 현행 디렉토리 안에 있는 특정 화일을 지정한다면 다음과 같이 한다.

```
A>ERASE BUDGET88.WKS
```

DOS는 현재 작업중인 A드라이브에서 BUDGET88.WKS라는 화일을 찾고 등록부에서 그 이름을 지워버린다. 그러나 다음 명령을 보면

```
B>ERASE A:\LOTUS\*.WKS
```

DOS는 이 명령을 해석할 때 A드라이브에 있는 화일에 관련된 명령이라고 생각하게 된다. 더욱 중요한 것은 해당 화일이 루트디렉토리 안에 있는 LOTUS라 불리우는 서브디렉토리에 있는 화일이라는 점이다. (A : 다음에 나오는 최초의 \는 루트디렉토리를 나타내고 뒤에 나오는 \는 디렉토리 구분용이다.) 위의 경우는 하나의 화일만 지우는 것이 아니라 .WKS라는 확장자를 가지는 화일들을 모두 지운다. 이제는 왜 DOS가 화일이름 명세에 대하여 중요하게 생각하는지 알았을 것이다. 왜냐하면 틀린 서브디렉토리나 틀린 화일을 입력했을때 중요한 화일을 지워 버릴 수가 있기 때문이다.

일반적으로 내용에 어울리는 단어로 화일이름을 택하면 후에 관리하기가 편리해진다. 예를 들어 어떤 학급의 성적표가 들어 있는 화일을 ABC라는 이름으로 지정하는 것보다 TEST, 또는 SUNGJUK이라는 이름으로 지정하는 것이 더 낫지 않겠는가?

더 나아가 함축성있는 의미로 화일이름을 만드는 경우도 필요하다. 화일이름은 단지 8문자 길이만 가질 수 있기 때문에 이보다 더 긴 단어는 함축성있게 축약시켜야 하기 때문이다. 그리고 여기서 함축성있다는 말은 제한된 문자수로 원래의 의미를 쉽게 기억하고 쉽게 재생시킬 수 있어야 한다는 말이 된다.

다음 예들을 생각해 보자.

FIST (Financial Statement)
QTFIREP (Quartely Financial Report)
BK3 (Chapter 3 of a book manuscript)

이런 이름들을 FINSTM, QTFINPRT, BOOKCH3라고 확장해서 썼다면 다른 이용자들에게 더 쉽게 이해를 줄 수 있을 것이다. 즉 이렇게 되면 전체 화일의 집합을 몇개의 소 부문들로 나누는 응용작업도 가능한데 1988년 1차분기에 대한 A부서의 예산을 작성한다고 가정하자. 이 경우 화일관리 작업을 많이 해 본 독자는 BUDGA881.WKS와 같이 이름을 할당하고 싶을 것이다. 처음 4자는 예산화일(BUDGET)이라는 의미이고 5번째의 A는 A부서, 그리고 마지막 881은 1988년 1차 분기임을 의미한다.

화일 이름은 8문자라는 고정된 위치내에 정보가 코드화되기 때문에 DOS의 SORT명령을 이용하면 디렉토리의 목록을 분류할 수도 있다. 이런 측면을 고려해 화일이름을 BUDGA188.WKS라고 정하는 것이 적합하지 않다는 것을 알 수 있다. 왜냐하면 SORT를 하려면 연도 앞에 분기를 쓰는 것보다 오히려 분기 앞에 년을 써야만 적당히 분류가 가능하기 때문이다. 만일 연도 앞에 부서를 쓰는 것보다 부서 앞에 연도를 쓰기 원한다면 BUDG881A.WKS 로 화일이름을 바꾸면 된다.

다음은 화일 확장자명에 대해서 알아 보자. 3문자까지 가능한 확장자명은

시스템 소프트웨어가 자동적으로 만들어 주는 경우가 있는가 하면 사용자가 직접 지정해 주는 경우가 있다. 한 예로 LOTUS 1-2-3은 모든 워크시트 화일들을 ".WKS"로 확장자명을 자동적으로 만든다. 그러나 다른 경우에는 사용자 스스로가 임의대로 확장자명을 결정할 수 있는데 대개는 화일의 일반적인 형태를 지시하는 의미로 확장자명을 쓰고 있다. 즉 텍스트 화일인 경우는 ".TXT"로, 데이터 화일인 경우는 ".DAT"로 사용되고 있다. 그러나 확장자명 중에 .EXE, .COM, .SYS 등은 DOS나 응용 프로그램에 의해 이미 예약되어 있기 때문에 이들 확장자명 만큼은 임의대로 쓸 수 없다는 것을 기억하자. 보통 DOS상태에서 실행가능한 프로그램들은 .EXE 또는 .COM 확장자명을 가진다. 대부분의 DOS프로그램 화일들은 .COM화일들이다. 또한 배치화일일 경우는 .BAT확장자를 가져야 하며 이렇게 되어야만 DOS가 인식하고 배치처리 루틴으로 실행을 시킨다. 그리고 각 프로그래밍 언어들도 자신들의 고유한 확장자명을 가지고 있다. 쉽게 볼 수 있는 예들이, 베이직 프로그램인 경우 .BAS, 터보 파스칼인 경우 .PAS, 어셈블리 프로그램인 경우는 .ASM, 알엠코볼인 경우 .CBL 등이다. 때로는 프로그램 실행시 스크래치 시트같은 일시적인 작업화일을 필요로 할 때도 있다. .$$$확장자명이 바로 이런 경우에 쓰이며 보통은 프로그램 실행이 끝나기 전에 지워져 버린다. 또 어떤 응용프로그램 작업시 프린터형태 출력을 만들고 그것을 한 화일처럼 보관할 필요성이 있는데 LOTUS 1-2-3에 의해 .PRN확장자가 붙는 화일들이 이 경우에 속한다. 프린터 데이터화일들은 표준 ASCII 형태로 저장되어 있어서 워드프로세싱 프로그램들에 의해 접근이 가능하게 되어 있다. 워드스타같은 워드프로세싱 프로그램들은 자동적으로 화일들의 백업복사를 만들며 .BAK란 확장자명을 만들어 낸다.

□주의
만일 8문자보다 긴 화일이름을 지정했거나 3문자보다 긴 확장자명을 지정하면 DOS는 나머지 문자를 무시해 버린다.

□주의
화일이름으로 특정 주변기기 이름을 사용하지 않도록 주의하라. 디바이스(장치) 이름들은 프린터같은 특별한 하드웨어 디바이스들을 위한 이름이기 때문이다. 이런 예약된 디바이스 이름들은 그들이 참조할 장치를 뜻한다.

예를 들면 다음의 경우이다.

- AUX : COM1과 같으며 최초의 시리얼 포트
- COM# : 비동기 채널(#1 또는 #2), 시리얼 포트
- CON : 콘솔(디스플레이 스크린)
- LPT# : 병렬 프린터 포트(#1 또는 #2)
- NUL : 널디바이스, 메모리에서 일시적으로 잡아두는 공간
- PRN : 최초의 병렬 프린터 포트

사용자들은 이런 이름들을 가지고 임의적으로 화일을 만들어서는 안된다.

□주의
화일을 복사하고 싶을 때는 COPY명령어를 이용하면 된다.

COPY명령어

COPY명령어는 DOS명령어 중 가장 많이 쓰이는 명령어에 속한다. 이 명령을 이용하면 하나의 디스크에서 다른 곳으로 하나 또는 그 이상의 화일들을 복사할 수 있다. 그리고 복사를 하려면 두개의 화일이름이 필요한데 하나는 복사하려는 원본(소스화일)의 이름이고 하나는 복사되는 새로운 복사본의 이름인 것이다. 예를 들어 A드라이브에 있는 BUDGET88.WKS라는 화일을 B드라이브에 같은 이름으로 복사한다고 하자.

```
A>COPY BUDGET88.WKS B:
```

유사한 경우로 만일 다른 드라이브에서 현재 지정드라이브로 화일을 같은 이름으로 복사하려고 한다면 COPY명령을 다음과 같이 쓴다.

```
A>COPY B:INVEST.WKS
```

이 명령에 의해 B드라이브에 있는 INVEST .WKS라는 화일이 A드라이브에 같은 이름으로 복사된다.

□주의
COPY명령을 쓸때 우선 출발 파라미터를 쓰고 목적 파라미터를 쓴다.

COPY명령어를 이용할 때 발생하는 에러의 대부분은 출발지와 목적지 화일이름의 순서를 거꾸로 지정한 경우이다. 특히 다른 운영체제와 친숙하게 사용하던 사용자들에게는 더욱 그렇다. (예를 들면 CP / M에서 화일을 복사할때는 목적지를 먼저 쓰고 출발지를 뒤에 쓴다.)

이러한 실수를 막기 위해서는 자주 실습해 보는 것이 제일 좋다. 또는 이렇게 한번 생각해 보자. 화일을 복사하는 것은 책을 한쪽 탁자에서 다른 탁자로 옮기는 것과 같다. 누구든 책을 옮기기 전에 먼저 그 책을 집어 들어야만 한다. 그래서 이론적으로도 출발지를 먼저 결정해야 한다.

□주의
COPY명령을 이용할 때 작업 대상 디스켓은 반드시 포매팅되어 있어야 한다.

쓰기방지 테이프가 붙어 있으면 COPY명령을 사용해도 그 디스켓에는 아무것도 씌여지지 않는다. 쓰기방지 테이프는 주로 디스켓속의 원본 화일이 파괴되는 것을 막기 위해 쓰인다.

비록 COPY명령이 다재다능하고 막강한 기능을 가지고 있지만 잠재적인 위험성도 가지고 있다. 그러나 사용자 자신의 원본 화일에 겹쳐 쓰는 것을 불가능하게 함으로써 이 위험을 줄일

수 있다. 디스켓의 오른쪽 상단을 보면 작은 사각형 모양의 홈이 있는데 만일 그 부분을 쓰기방지 테이프로 막아버리면 디스켓은 읽을 수 있으나 쓸수는 없게 된다.

□주의
COPY명령의 경우, 소스화일에 확장자명이 있을 때 이를 명시해 주어야 한다.

예를 들어 BUDGET.WKS라는 화일을 복사할 때 확장자명을 지정해 주지 않으면 다음과 같이 실패로 돌아가게 된다.

```
A>COPY BUDGET B:
    0 File(s) copied
```

즉 디스켓에는 BUDGET.WKS라고 이름이 명시되어져 있기 때문에 DOS는 BUDGET라는 이름의 화일은 찾을 수 없다고 대답한다. 그러나 소스화일의 확장자명은 명시했으나 목적화일을 쓸때 빠뜨렸다면 확장자명이 없는 이름으로 COPY가 이루어 진다.

```
A>COPY BUDGET.WKS B:BUDGET
    1 File(s) copied
```

그리고 하나의 플로피디스크와 하드디스크만을 가지고 있는 경우라도 한 디스켓에서 다른 디스켓으로 화일의 복사가 가능하다. 만일 두개의 플로피디스크를 가지고 있다면 화일의 복사는 더욱더 쉽다.

```
C>COPY A:*.WKS  B:
```

다음, 더욱 효과적인 방법은 당신의 하드디스크에 있는 일시적인 디렉토리를 이용하는 것이다. 즉 하드디스크상에 일시적인 디렉토리를 만들어 놓고 필요한 화일을 A드라이브로부터 읽어 들인다. 그리고 나서 다시 드라이브에 디스켓을 바꾸어 끼고 하드디스크의 일시적인 디렉토리안에 있는 화일을 플로피디스크로 복사하면 된다. 사용자는 이러한 작업을 용이하게 하기 위해 배치화일을 이용할 수 있다(리스트 1 참조).

마지막 줄의^Z는 여러분의 화일작성이 끝났음을 알리고 디스크에 그것을 저장하라는 제어 명령이다.

COPY명령어의 응용

□주의
복사의 형태에 관계없이 COPY명령어는 지정된 목적화일이름이 이미 디스켓에 존재하든지 안하든지 간에 실행한다.

만일 디스켓속에 목적화일이름과 똑같은 화일을 가지고 있지 않다면 COPY 명령어를 실행했을때 새 화일이 만들어 진다. 만일 디스크에 이미 존재하는 화일이름의 화일을 만들었다면 그 화일위에 겹쳐져서 쓰이게 될 것이며 이렇게 될 경우 중요한 정보(기존의 내용)가 파괴되어 버릴 수도 있다.

DOS는 사용자에게 이미 존재하는 화일이 있으니 조심하라고 경고하지 않는다. 그러므로 화일을 복사할때 먼저 지정될 화일의 이름과 같은 화일이 존재하는지 조사해 보고나서 실행하는 습관을 가지도록 하자.

□주의
디스크 공간이 불충분한 상태에서 COPY명령으로 구버전의 화일을 새로이 수정하려고 했을때 목적지 드라이브에 있는 구버전은 그 과정에서 지워지게 된다.

화일의 수정되었을때 구버전은 디스크에 복사되는 새 버전의 공간을 만들기 위해 지워져 버린다.

DOS가 새 버전을 복사하려고 할때 전체화일을 충당하기 위한 디스크 공간이 충분하지 않다면 실행이 실패로 돌아가게 된다.

다음 예를 보자.

```
A>COPY NOTEFILE.BAK B:
 Insufficient disk space
      0 File(s) copied
```

만일 목적디스켓의 디렉토리를 살펴보면 NOTEFILE.BAK화일이 복사되지 않았음을 볼 수 있을 것이다. 다시 화일을 저장하고 싶다면 또 다른 포맷된 디스켓을 사용하거나 그렇지 않다면 필요없는 화일을 지운다.

특정 디스켓에 있는 화일의 목록을 보여주는 DIR명령어를 사용하면 화일의 목록과 함께 얼마나 많은 바이트들이 사용가능한가를 마지막 줄에 표시해주므로 이 잔여용량을 확인하고 작업을 하면 좋다.

예)

```
A>DIR

 Volume in drive A has no label
 Directory of A:\

FILE1  WKS   54224 2-08-87  7:37a
FILE2  WKS   81093 2-10-87 11:08p
FILE3  WKS   91886 3-22-87 10:09a
FILE4  WKS   67917 4-03-87 20:19p
FILE5  WKS  101220 5-01-87 12:33p
FILE6  WKS   49105 5-01-87 17:21p
    6 File(s)     8961 bytes free
```

사용가능한 용량이 9000바이트 정도라면 이 디스켓은 거의 꽉 찬 상태로 보아야 한다.

□주의
와일드카드를 COPY명령과 함께 쓸 경우도 복사하려는 화일들에 비해 지정디스크에 공간이 충분하지 않다면 용량이 허락할 때까지 복사되고 그 나머지 화일은 복사가 안된다.

```
A>COPY *.WKS B:
  BUDGET.WKS
  INVEST.WKS

  Insufficient disk space

        2 File(s) copied
```

공간이 부족한 탓으로 위 명령은 .WKS를 가진 화일 중 두 화일만 복사가 된 것이다.

같은 디스크의 같은 부디렉토리안에

〔리스트 1〕

```
A>COPY CON: COPYFILE.BAT
ECHO OFF
ECHO COPIES FILES FROM ONE
ECHO DISKETTE TO ANOTHER.
ECHO OK TO ERASE TEMPDIR ON C: ?
ECHO PRESS ANY KEY - ELSE CTRL+BREAK
PAUSE
ERASE C:\TEMPDIR\*.*
ECHO INSERT SOURCE DISKETTE IN DRIVE A:
ECHO PRESS ANY KEY WHEN READY
PAUSE
COPY A:*.* C:\TEMPDIR
ECHO REMOVE SOURCE DISKETTE FROM A:
ECHO INSERT DESTINATION DISKETTE IN DRIVE A:
ECHO PRESS ANY KEY WHEN READY
PAUSE
COPY C:\TEMPDIR\*.* A:
^Z
```

저장된 화일들은 상이한 화일 이름들과 확장자를 가져야 한다. 그러나 서로 다른 디렉토리에서는 같은 화일 이름이라도 존재할 수 있다.

□주의
화일들을 어떤 조건하에 선택적으로 복사하고 싶을때 와일드카드 문자를 이용하면 편리하다.

와일드카드 문자는 많은 DOS명령어들과 함께 사용될 수 있는데 특히 COPY명령어와 함께 쓸때는 주의해서 써야 한다. 왜냐하면 한 명령어로 여러개의 화일들을 복사해 버리기 때문이다.

*문자는 모든 화일이름과 확장자를 대체하지만 ?문자는 한 화일이름에서 한 문자만을 상징한다.

```
A>COPY *.* B:
```

이 명령어는 현재 A드라이브의 부디렉토리에 있는 모든 화일을 B드라이브로 복사하라는 명령이다.

```
A>COPY B:*.COM
```

이 명령어는 .COM이라는 확장자를 가진 B드라이브에 있는 모든 화일을 A드라이브로 복사한다.

```
A>COPY ??BUDGET.WKS B:
```

이 명령어는 화일이름이 총 8문자로 구성되어 있으며 마지막 6문자로 BUDGET 이란 문자를 가지고 .WKS란 확장자를 가진 모든 화일을 복사한다. 키보드로부터 디스크화일로 정보를 복사하기 위해서는 다음과 같은 명령을 사용한다.

```
COPY CON: A:MEMO.DOC
```

화일의 끝이라는 표시(^Z)가 나타날 때 까지 키보드로 치는 모든 내용들이 MEMO.DOC라는 이름의 화일로 입력되어 A드라이브에 저장된다.

□주의
원본디스켓에 섹터단위로 흩어져 있는 화일을 COPY명령을 사용함으로써 새 디스켓에 깨끗이 정렬된 화일을 얻을 수 있다.

화일을 저장하고 삭제하는 과정에서 무질서한 공간이 만들어 진다. 즉 한 화면을 축적하기에는 충분하지 않은 공간들이 디스크 전체에 뿔뿔이 흩어진 상태로 존재하게 되고 복사되는 화일은 이 공간들을 찾아가며 서로 분리된 채 기억되어 진다.(그렇지만 논리적으로는 서로 연결되어 있다.)

이와 같이 화일들이 서로 인접하지 않은 위치에 저장되어 있을 때는 액세스시간이 오래 걸리며 이것은 응용프로그램과 마찬가지로 DOS작동에서도 성능을 둔화시켜 버린다. 또한 사용자들이 사용할 수 있는 전체 디스크 공간도 감소시키게 된다.

COPY *.*명령은 한 디스크에 있는 현재 부디렉토리에서 다른 곳으로 모든 화일을 효과적으로 복사한다.

□주의
여러개의 화일을 결합하는데도 COPY명령을 이용할 수 있다.

종종 여러개의 화일들을 결합하여 하나의 화일로 만들어야 하는 경우가 생긴다. COPY명령어의 결합명령은 처음부터 끝까지 뒤따르는 화일들을 결합하여 새로운 한 화일로 복사해 준다. 예를 들면 다음과 같다.

```
A>COPY FILE.TXT+OTHERFILE.TXT NEWFILE.TXT
```

위 명령은 두개의 화일 즉 FILE.TXT와 OTHERFILE.TXT란 화일을 결합하여 NEWFILE.TXT라는 새로운 한 화일로 복사해 준다.

DOS는 복사될 때 각각의 소스화일들의 이름을 화면에 표시하고 마지막에는 하나의 화일로 복사되었다고 화면에 표시해줄 것이다.

즉 다음과 같은 결과를 보게 된다.

```
A>COPY FILE1.TXT+FILE2.TXT HUGEFILE.TXT

        FILE1.TXT
        FILE2.TXT

                1 file(s) Copied
```

위의 경우는 동일 디스켓내에서 복사가 이루어졌기 때문에 목록을 살펴보면 소스화일들과 새로 만들어진 화일을 볼 수 있다.

```
A>DIR
 Volume in drive A has no label
 Directory of A:\

FILE1     TXT  11420  3-02-87  8:30a
FILE2     TXT  13476  3-05-87 10:33a
HUGEFILE  TXT  24896  3-10-87  9:21a
     3 File(s)  311,468 bytes free
```

이 결합기능은 사용자가 두개의 배치화일을 결합하고 싶을 때 특히 유용하게 사용될 수 있다.

□주의
COPY명령의 결합기능은 소스화일들이 아스키 화일(ASCII File)이어야 한다. 만일 이진 화일(Binary File)이라면 특별한 스위치 설정과 함께 써야 한다.

COPY명령은 /A와 /B라는 스위치변수를 갖고 있다. 비록 드물게 사용되기는 하지만 화일을 결합하기 위한 COPY명령인 경우 중요하게 쓰인다. ASCII화일을 A, 이진화일을 B로 표시하고 있으며 ASCII화일은 각종 문자를 포함하게 되며 이진화일들은 종종 비편집 문자 성격인 이진화일들을 저장한다. 보통 이진화일들을 결합하려고 하는 경우가 많지는 않지만 사용할 때는 꼭 /B옵션을 사용해야 한다.

만일 /A나 /B 중 아무 것도 쓰지 않았다면 DOS는 스스로 /A라고 가정한다.

다음과 같은 화일 디렉토리가 있다고 하자.

```
JAN.ACC
FEB.ACC
MAR.ACC
QTR.ACC
YTD.ACC
```

이러한 소스 화일들을 QTR.ACC라고 불리는 목적화일 하나로 병합시키려 한다고 하자.

이런 경우는 소스화일들을 모두 지정하는 번거로움 없이 다음과 같은 형태로 명령어를 쓰면 된다.

```
A>COPY *.ACC QTR.ACC
```

이 명령을 실행하는 동안 DOS는 다음과 같은 에러메세지를 나타낸다.

```
Contents of destination
lost before copy
```

이것은 DOS가 목적화일이름과 동일한 소스화일이름을 만났기 때문이다. 와일드카드가 화일이름으로 사용되었을 때 DOS는 와일드카드이름과 맞는 것을 찾을 때까지 모든 화일들을 훑어 내려간 다음 그 화일을 처리한다. 처음 화일 JAN.ACC는 ＊.ACC와 맞고 QTR.ACC란 목적화일에 포함이 된다. DOS는 실제로 QTR.ACC란 새 화일을 만들고 그것은 와일드카드 이름과 어울리는 모든 소스화일들에 대해 목적화일이 된다.

소스화일 QTR.ACC에 저장된 정보는 DOS가 같은 이름으로 목적화일을 만들 때 지워진다. 왜냐하면 같은 이름을 가진 화일들이 같은 디스크나 같은 서브디렉토리에 존재할 수 없기 때문이다. 소스화일 JAN.ACC, FEB.ACC 그리고 MAR.ACC는 새롭게 만들어진 QTR.ACC로 성공적으로 복사된다.

그러나 DOS가 QTR.ACC라는 이름의 화일을 첨가하려고 했을 때 원본 소스화일이 변경되었기 때문에 에러메시지가 나온다. 이때 원래 소스화일의 내용을 복구하기에는 너무 늦어버린다. 그러나 사실상 이런 상황에서 화일이 성공적으로 결합하도록 COPY명령어를 사용할 수 있다.

```
A>COPY QTR.ACC+*.ACC
```

바로 앞에 ＋표시가 없는 화일은 QTR.ACC이므로, 이것이 바로 목적화일이 된다.

QTR.ACC는 또한 처음의 소스화일 목록이 되며 이름이 동일하기 때문에 DOS는 이 처음 소스화일을 제외해 버리고 다른 모든 .ACC화일들을 목적화일에 첨가하게 될 것이다.

□주의
COPY명령어로 DATE와 TIME을 변경할 수 있다.

화일이 복사될 때 원래의 날짜와 시간도 같이 복사된다. 이러한 사항들을 변경하기 위해서는 COPY명령어를 다음과 같이 사용한다.

```
A>COPY B:FILE.EXT+
```

DOS는 FILE.EXT인 모든 화일(이 경우는 단지 하나의 화일)들을 결합하여 A드라이브에 하나의 화일로 그것을 저장할 것이다. DOS는 새로운 화일이 생성될 때 새로운 날짜와 시간을 세트시켜 준다.

보통 시간과 날짜를 변경하려는 이유는 컴퓨터를 부팅했을 때 시간과 날짜가 부정확하게 설정되었기 때문일 것이다. 만일 한 디스켓에서 다른 곳으로 화일을 옮기지 않고 시간과 날짜를 변경하려고 한다면

```
A>COPY B:FILE.EXT+,, B:
```

이란 명령을 쓰면 된다. 플러스 기호 뒤의 두개의 콤머는 DOS에게 플러스 뒤에 다른 화일이름을 찾지 말라고 지시하는 것으로써 프로그램화일이나 이진데이터 화일에서 같은 기능을 수행하려면 /B 스위치를 첨가한다.

```
A>COPY/B FILE.EXT+,,
```

또는

```
A>COPY FILE.EXT/B+,,
```

사용자는 또

```
A>COPY CON:"File name"
```

의 구문을 이용하여 키보드로 바로 입력하여 화일을 작성할 수도 있고 EDLIN을 사용하여 편집할 수도 있다.

□주의
COPY명령어는 디바이스들 사이에서 화일을 전달하는데도 사용할 수 있다.

TYPE명령은 스크린에 특정 화일의 내용을 표시해 주며 TYPE명령을 쓰기전에 <CTRL+P> 키를 누름으로써 이후의 출력을 프린터로 보낼 수 있다.

TYPE명령어는 동시에 한 화일을 다룰 수 있기 때문에 TYPE *.TXT라는 구문을 쓸 수 없다. 그럼 현재 디렉토리에 있는 모든 화일을 프린트하거나 화면에 표시해 보고 싶을 때는 어떻게 할까?

TYPE명령은 와일드카드 문자를 받아 들일 수 없으므로 다음과 같은 메시지를 출력한다.

```
File not found
```

이런 경우도 COPY명령을 쓰면 된다. 즉 모든 화일을 화면에 복사하라는 의미로

```
COPY *.TXT CON:
```

이란 명령을 쓴다.

또는 프린트하려면

```
COPY *.TXT PRN
```

을 쓴다.

□주의
여러분은 DOS에서 바로 타이프라이터처럼 키보드로 쳐서 프린터로 인쇄할 수 있다.

COPY CON: LPT1:(LPT1은 병렬 프린터 #1을 의미)이란 명령을 쓰면 키보드에서 친 모든 입력이 바로 프린터로 보내진다. 끝마쳤을 때 <F6> 함수키(^Z)를 치고 처리의 끝을 알리는 <ENTER>키를 치면 된다.

□주의
만일 디스크에서 화일을 지우려면 ERASE나 DELETE(DEL) 명령을 쓴다.

```
ERASE *.COM
```

그러면 .COM이란 확장자를 가진 모든 화일이 지워진다.

DEL *.COM도 마찬가지 효과를 얻는다. ERASE ??BUDGET.WKS라 쓰면 마지막 '6자로 BUDGET이란 이름을 가진 총 8문자의 화일로써 .WKS 확장자를 가진 모든 화일을 지운다.

□주의
ERASE(DEL)은 경고문없이 처리하기 때문에 잘못 실수로 화일을 지우기 쉽다.

와일드 카드문자로 모든 화일을 지울때 지우려하지 않았던 화일도 지워버릴 수 있으므로 매우 위험하다. 이런 경우 DOS는

```
Are you sure (Y/N)?
```

라는 경고문을 표시하여 실수를 방지하려고 한다. 그러나 이런 것은 전체 불행에 대한 충분한 방지책이 되지는 못하므로 ERASE명령을 쓰기 전에 항상 DIR명령으로 디렉토리의 목록을 보고나서 삭제할 화일을 결정하는 것이 좋다.

□주의
서브디렉토리에 있는 모든 화일을 지우려 하지 않는 한 ERASE명령에 서브디렉토리 이름을 쓰지 않도록 주의하라.

만일 루트 디렉토리 밑에 TEST란 서브디렉토리를 갖고 있다고 가정하자. ERASE \TEST라고 쓰면 서브디렉토리에 있는 모든 화일이 지워진다. (이 경우 서브디렉토리 이름 자체는 남아 있다. 참고로 서브디렉토리를 지울때는 DOS의 RD명령을 이용한다.)

□주의
두 화일이 동일한가를 알아보기 위해 COMP 란 명령을 사용한다.

서로 다른 시간에 만들어진 두개의 동일한 화일을 비교하기 위해 이 명령을 사용할 수 있다. 한 통신선을 통해 전달된 화일이 정확히 되었는가를 검토하기 위한 방법은 그것을 다른 이름으로 두번 전달하고 난후 두 화일을 COMP명령으로 비교하면 된다. 만일 같지 않다면 전달이 잘못된 것이다. 명령 형식은

```
COMP filename1.ext filename2.ext
```

이다.

화일이름 확장자는 꼭 명시해야 한다. 그렇지 않으면 비교할 화일을 찾지 못할 수 있기 때문이다. 비교가 성공적으로 끝나면 DOS는

```
File Compare OK
```

화일이 동일하지 않았을 때 DOS는 두 화일에서 서로 맞지 않은 정보들을 포함하고 있는 각 부분에 대해 에러메시지를 나타낸다.

□주의
COMP과정을 간단히 하기 위해 와일드카드를 사용할 수 있다.

```
COMP A:*.* B:
```

이 명령은 동일하게 이름지어진 화일들에 대해 두개의 디스크를 체크하라고 DOS에게 명령한다.

```
COMP A:*.EXE B:*.BAK
```

B드라이브에 같은 이름을 갖고 .BAK 확장자를 가진 화일과 A드라이브의 .EXE 확장자를 가진 각각의 화일을 비교할 것이다.

□주의
화일들은 비교하기 위해서는 같은 길이의 화일을 지정해 주어야 한다.

COMP명령은 한 화일이 다른 것보다 크다면 작업할 수 없다. 동일한 길이를 갖지 않은 화일을 비교하려하면 DOS는 Files are different sizes란 메세지를 나타낸다.

□주의
RENAME(REN)명령은 하나 또는 그 이상의 화일이름을 바꿀 때 쓰인다.

□주의
RENAME은 화일의 위치를 변경하지 않기 때문에 목적드라이브 지정자는 무시

되고 만일 디스크에 이미 존재하는 화일이름을 명시했다면 명령은 실패하게 된다.

만일 한 화일을 Rename시키고 다른 디스크나 부디렉토리로 옮기고 싶다면 COPY명령을 쓴다.

□주의

만일 하나의 디스크에 같은 이름이나 확장자를 가진 하나 또는 그 이상의 화일이 있다면 여러분은 한 그룹으로써 이러한 화일들을 Rename시키기 위해 와일드카드 문자를 이용할 수 있다.

```
A>RENAME ACCOUNTS.* SALES.*
A>RENAME *.DOC *.TXT
```

□주의

와일드카드 문자를 이용하여 Rename시키기 전에 하드디스크에 있는 임시 디렉토리나 다른 디스켓에 모든 화일을 복사해 두는 습관을 갖는 것이 좋다.

그러나 화일을 성공적으로 Rename시킨 후에는 이 임시 디렉토리에 있는 내용을 지워도 무방하다.

□주의

응용 프로그램에서 사용된 화일들의 이름은 변경하지 않는 것이 좋다.

예를 들어 워드프로세싱 프로그램인 워드스타는 WSOVLY1.OVR이란 이름의 화일에 접근할 수 없다면 동작이 불가능하다. 만일 이런 화일의 이름이나 확장자를 변경한다면 워드스타는 새로운 이름을 인식못하고 동작을 못하게 된다.

화일에서 지워진 데이터는 아직 디스크에 존재하고 "Un–erase"프로그램에 의해 복구될 수 있다.

지워진 화일이 복구될 수 있다는 말이 이상하게 들릴지 모르지만 사실상 화일이 지워졌을 때 데이터는 파괴되지 않고 표준 DOS명령으로 접근하지 못하도록한 것 뿐이다. 만일 종이의 내용을 제거하기 위해 종이를 접어서 쓰레기통에 버렸다고 하자. 만일 그 내용이 다시 필요하다면 쓰레기통을 비우지 않은 한 종이의 내용을 다시 꺼내 재생시킬 수 있다. "Un–erase" 프로그램은 이런 원리를 이용하여 같은 위치에 저장된 다른 화일이 없다면 여러분의 데이터를 복구할 수 있는 프로그램이다.

88년 12월호 퍼스널컴퓨터에 소개된 UNDEL.COM이나 노턴 유틸리티 디스켓에 포함되어 있는 QU.EXE 프로그램 등이 그것이다.

□주의

VERIFY명령은 디스크에 쓰기 작업을 했을 때 데이터가 정확히 기록되었는지 검토하는데 이용되곤 한다.

VERIFY명령이 실행될 때 디스크 드라이브 활동이 느려질 수 있다.

이것은 DOS오퍼레이션과 응용 프로그램에 영향을 준다.

만일 DOS가 모든 디스크 쓰기 과정을 VERIFY한다고 한다면 많은 DOS 명령과 응용프로그램들은 신중하게 더 천천히 작동할 것이다. 일단 VERIFY가 작용하면 여러분이 시스템을 다시 시작하거나 끄지 않는한 그 영향력이 계속된다. PC

〈용어해설〉

●디렉토리(Directory)
시스템이 갖고 있는 화일의 일람표로 화일의 명칭, 위치, 일자 등이 저장된다. 이 디렉토리는 화일관리의 중심이 된다.

●패스(Path)
그래프에 있어서 어떤 마디에서 다른 마디에의 경로를 나타내는 가지 또는 마디의 열

●와일드 카드(Wild Card)
화일의 이름과 그 확장자를 지정할 때 사용되며 ?와 *의 2개 특수 문자를 지칭한다.

●드라이버(Driver)
주변장치나 그 인터페이스를 CPU에서 제어하는 작은 프로그램

●시스템 소프트웨어(System Software)
컴퓨터 시스템을 구성하는 소프트웨어로 오퍼레이팅 시스템(OS)와 동의어이다.

●텍스트화일(Text File)
프로그램 실행을 위한 화일이 아닌 전달해야 할 정보를 포함한 메시지와 메모 등을 담은 화일(예를 들면 배치화일이나 워드프로세서로 작성한 화일).

●배치화일(Batch File)
도스가 한번에 하나씩 실행하는 하나 또는 그 이상의 명령어를 갖는 화일.

●워드프로세서(Word Processor)
영문이나 한글의 문서를 변경, 추가, 편집, 교정하여 인쇄출력을 내기 위한 문장편집 시스템. CRT상에서 편집한 화면을 그대로 프린터에 출력하거나 플로피디스크에 저장하는 기능을 갖는다.

●응용 프로그램(Application Program)
유저의 업무를 처리하기 위해 메이커 또는 유저에 의해서 작성된 프로그램.

●백업복사(Back-up Copy)
화일이나 데이터 세트의 복사를 말하며 원래의 화일이나 데이터 세트가 파괴된 경우에 대비하기 위한 것.

●파라미터(Parameter)
프로그램이나 서브 시스템 사이의 인터페이스를 취하기 위한 입출력 변수나 상수를 말한다.

●버전(Version)
어떤 시스템을 디버그하거나 또는 어떤 목적을 갖고 변경하여 완성된 시스템으로 예를 들면 버전에 번호를 붙여 디버그시마다 그 번호를 늘린다. 따라서 버전 1을 갱신한 시스템을 버전 2나 1.5로 한다. 이와 같은 버전의 갱신을 버전업이라고 한다.

허클리스 그래픽카드

안대혁 /삼보컴퓨터 기술연구소

허클리스카드는 대부분의 사람들이 가장 많이 사용하면서도 이의 하드웨어나 소프트웨어적인 제어방법에 대해서는 잘 알지 못하고 있는 실정이다.

허클리스 카드의 소개

지금 우리나라에서 IBM-PC 호환기종을 사용하는 사용자의 대부분은 허클리스 그래픽스카드를 사용하고 있을 것이다. 1982년 미국의 Hercules Computer Technology사에서 판매하기 시작한 허클리스 그래픽카드(이하 허클리스라 칭함)가 표준 그래픽디바이스로 정착한 것은 1982년 말이다. 그후 대부분의 소프트웨어 회사에서는 허클리스를 지원하는그래픽용 디바이스 드라이버를 작성하여 판매하기 시작하였다. 이때부터 모노크롬 그래픽의 표준 그래픽카드로 허클리스가 득세하기 시작하였다. 그 원인은 컬러모니터의 가격이 비싸고 또한 컬러 그래픽카드의 가격이 그에 상응하였기 때문이다. 우리나라에서도 허클리스가 그래픽과 비디오카드의 표준으로 정착한 것은 위와 같은 이유와 몇가지 이유가 있었다.

그 이유로는 청계천의 컴퓨터 판매업체들이 컬러모니터를 구하기 어려웠으며 가격을 내리기 위한 정책이 거기에 더해졌고, 가장 큰 이유로는 1983년에 국내에서 최초로 개발된 한글카드(한도엔지니어링 개발, 제일정밀판매)가 허클리스 카드를 이용하여 만들어졌기 때문이라고 생각된다. 어찌되었건 우리나라에서의 표준 비디오 / 그래픽 카드로는 허클리스가 사용되고 있고 또한 대부분(90% 이상)의 사용자들이 쓰고 있는 것만은 사실이다.

일부 대기업체들에서는 허클리스가 아닌 CGA나 MDA 혹은 MCGA를 사용하고 있으나 호환성의 결여(?-허클리스와의)로 수행되지 않는 프로그램들이 생겨나고 이로 인해 여러 사용자들이 골치를 썩고 있는 실정이다. 몇가지 예를 들면 삼성의 SPC 시리즈는 MCGA를 사용하고 금성의 PC-24는 CGA를, 또한 대우통신의 PRO 시리즈는 MGA를 사용하므로 이들 기종의 사용자는 무료로 배부되는(?) 보석글(사실은 THP나 NKP)을 사용하지 못하고 동사에서 판매하고 있는 전용의 한글 워드프로세서를 구입하여 사용해야 하는 어려움을 겪고 있다.

어찌 되었든 이들 문제는 업계의 방향이 이들 모두를 지원하는 새로운 비디오 카드를 생산하는 쪽으로 돌려졌기 때문에 종내에는 해결될 것이나 한동안은 여러가지 문제점들을 안고있을 것이다.

이야기가 잠시 옆으로 빗나간 것 같은데 이제 제자리로 돌아와서 허클리스에 대한 이야기를 계속하자. 과연 허클리스란 무엇인가에 대한 명확한 해답을 허클리스의 사용자 매뉴얼을 참조하여 적어보았다.

"허클리스 그래픽카드는 IBM-PC 모노크롬 디스플레이를 위한 고해상도의 그래픽카드이다. 허클리스는 IBM의 MDA(Momochrome Display / Printer Adapter)와 이를 위한 소프트웨어에 대치할 수 있는 것이다. 허클리스는 IBM의 고해상도 모노크롬 캐릭터 세트와 동일한 스타일의 문자형을 제공하며, 또한 프린터용의 인터페이스도 제공되고 있다. 허클리스는 IBM 모노크롬 디스플레이상에서 720(수평)*348(수직)의 해상도를 제공하며 그래픽스를 위한 소프트웨어도 제공된다. 또한 여러종류의 그래픽 프로그램에서 허클리스가 지원하는 것과 동일한 그래픽을 제공한다."결국 허클리스는 모노크롬 글자 / 그래픽 그리고 프린터 지원용의 인터페이스라고 밝히고 있다.

허클리스 카드의 조작

허클리스는 전형적인 비디오용 카드로서 CRTC(CRT 콘트롤러)로는 모토롤라사의 MC6845를 사용하고 있다. MC6845는 대부분의 비디오카드에 사용되고 있는 CRTC로서 이에 관한 설명은 조금뒤에 하기로 하겠다. 그리고 비디오램은 64K바이트를 사용하고 있으며(4164*8개), 라이트펜을 위한 회로 그리고 센트로닉스 호환의 프린터 콘트롤로직을 가지고 있다.

최근에 개발된 허클리스 그래픽카드 플러스는 프린터를 제어하는 LPT112라는 자사제작의 별도 IC를 사용하고 있으나 초기의 허클리스는 이를 사용하고 있지 않다.

(1) 소프트웨어 스위치

대부분의 사용자들은 'HGC.COM'이라는 프로그램을 알고 있을 것이다. 이 프로그램은 주로 THP나 NKP를 올리기 전에 먼저 실행시킨다. 이 HGC는 사용자의 사용의도에 맞게 하드웨어를 변경시키는 역할을 하고 있다. 소프트웨어 스위치란 하드웨어에 의한 스위치(예를 들어 점퍼나 딥 스위치등)가 아니라 소프트웨어로 이를 대치하는 작용을 해주는 것을 말한다. HGC에는 스위치제어용 세가지 옵션과 스크린을 제어하는 또다른 한가지 옵션이 있는데 이에 관하여 알아보자.

① DIAG 옵션

DIAG옵션은 IBM의 Diagnostic 프로그램과 Exploring 프로그램을 사용하기 위하여 사용되는 것이다. 허큘리스는 전원투입시 항상 이 상태로 세트된다. 만일 HALF나 FULL상태에서 이 모드로 돌아오려면 다음과 같다.

```
A>HGC DIAG [ENTER]
```

② HALF 옵션

HALF옵션은 IBM 컬러카드(종류에 관계없이)를 허큘리스와 동시에 사용하고자 할때 사용하는 옵션이다. 만일 IBM 컬러카드를 허큘리스와 동시에 사용하려면 반드시 HALF나 DIAG에 있어야 하며 FULL로는 지정하지 말아야한다.(잘못 지정하면 하드웨어적인 손상을 입을 수도 있다.)

③ FULL 옵션

FULL옵션은 허큘리스의 텍스트와 그래픽을 사용하는 경우에만 지정한다. 기억할 것은 위에서도 말했듯이 이 옵션을 컬러카드와는 사용하지 말아야 한다. 물론, 허큘리스 컬러카드의 경우에도 마찬가지이다. LOTUS 1-2-3과 HBASIC, NKP등이 이를 사용하는 대표적인 프로그램이다.

④ SAVE 옵션

모노크롬 모니터의 경우 한곳에 오랫동안 글자가 위치하고 있으면 화상에 손상을 받아 모니터의 수명이 줄어든다. 이를 보상하기 위하여 HGC 프로그램에는 몇분(정확히는 모르나 대략 5분정도인 것 같다.) 동안 키보드를 누르지 않으면 화면을 꺼주는 루틴이 있다(램상주이다). 그리고 아무키나 다시 누르면 원래의 화면이 나온다.

이와 비슷한 프로그램으로서 SRCN SAVE, MONO BLANK, MONBLK2, MONITOR BLANK등의 프로그램들이 있으나 이들은 절대로 허큘리스에서 사용하지 말아야 한다. 또한 컬러카드(허큘리스 컬러카드도 포함해서) 들에서는 HGC SAVE 또한 사용하면 안된다. 어드레스가 다르기 때문에 손상을 입을 우려가 있다. SAVE옵션은 위의 새 옵션과 동시에 사용할 수 있다. 이때는

```
A>HGC FULL SAVE
```

와 같이 사용하면 된다.

사실 위의 DIAG, HALF, FULL 세가지 옵션은 하드웨어의 어드레스에 특정의 데이터를 넣음으로써 생략할 수가 있다. NKP등이 이를 자체적으로 지원하지 못하는 예인데 이것은 사실 아주 간단하다. 이의 어드레스와 데이터를 소개하고 NKP를 고쳐서 HGC를 사용하지 않고도 프로그램을 올리는 방법을 소개한다.

(2) 소프트웨어 스위치의 직접조작법

먼저 어드레스와 데이터를 소개하기로 한다. 나중에 뒷부분에서 언급하겠지만 이 어드레스는 Configuration Switch 로서 출력만이 가능하다.

```
Address : 3BF
Data    : 0 {DIAG}
        : 1 {HALF}
        : 3 {FULL}
```

이를 베이직등의 고급언어에서 사용하려면 OUT명령을 이용하면 된다. 즉,

```
OUT &H3BF,0 [ENTER]
```

와 같이 하면 되고

어셈블러에서 사용하려면 아래와 같이 하면 된다.

```
MOV AX,3BFH
MOV DX,AX
MOV AL,03H ←해당옵션의 데이터
OUT DX,AL
```

이를 이용하여 NKP를 수정하면 아래와 같다.

```
A>DEBUG NKP.COM
-A100
33E6:0100 MOV AX,3BF
33E6:0103 MOV DX, AX
33E6:0105 MOV AL,03
33E6:0107 OUT DX,AL
33E6:0108 JMP 91B1
33E6:010B
-W
Writing 91FA bytes
-Q
A>
```

이후부터 NKP를 사용할 때 귀찮게 HGC FULL을 선언하지 않아도 된다.

(3) HBASIC에 관하여

HBASIC이란 허큘리스 카드를 구입하면 함께 따라나오는 소프트웨어로, 허큘리스에서 BASICA의 모든 그래픽 명령을 사용할 수 있게 해주는 프로그램이다. 이 디스켓안에는 몇가지의 화일들이 함께 따라 나오는데 이들은 다음과 같다.

```
HGC.COM
HBASIC.EXE
HBASIC1.EXE
PATTERN.BAS
FONT.COM
```

먼저 HBASIC을 구동시키려면 이들 프로그램이 있는 디스크에 BASICA를 복사해 넣는다. 그리고 나서 'HBASIC'이라고 쳐넣으면 된다. 이때 베이직안에서 캐릭터코드 128부터 255까지를 사용하고 싶으면 HBASIC을 올리기

전에 'FONT.COM'을 먼저 실행시키면 된다.

(4) HBASIC의 사양

① HBASIC은 아래와 같은 명령문들은 지원하지 않는다.

```
COLOR
SCREEN 0
SCREEN 1
WIDTH 40
```

② HBASIC은 가로 9 세로 14 도트의 문자를 사용하지만 BASICA는 8*8 도트를 사용한다. 따라서 문자의 위치와 그래픽의 위치가 CGA와는 다르다.

③ HBASIC은 720*348 도트의 해상도를 가짐으로 CGA에서(640*200)와는 가로 세로의 비례가 달라지게 되는데 이때에는 가로 세로 비율을 2/3으로 맞추어서 사용하면 된다. 예를 들어 원이 찌그러지지 않게 하려면

```
CIRCLE(360,174),100,2,,,2/3
```

등과 같이 사용하면 된다.

④ HBASIC은 정상적인 BASICA보다 수행속도가 느리다.(그래픽쪽의 처리때문이다.)

⑤ SCREEN 2만을 지원한다.

⑥ 따라서 IBM 컬러카드를 사용하여 작성된 프로그램을 HBASIC에서 수행시키려면 아래의 몇가지 점을 유의해서 고치면 된다.

- SCREEN 0나 SCREEN 1을 SCREEN 2로 고친다.
- COLOR 명령문을 모두 없앤다.
- WIDTH 40을 WIDTH 80으로 고침.
- 가로 세로 비율을 2/3로 고친다.
- x/y 좌표를 표시하는 부분을 좀더 크게 만든다.

허클리스 카드의 하드웨어 구성

이야기가 점점 딱딱한 것으로 흐르니 잠시 여담을 하나 소개 하기로 하겠다. 'HERCULES'란 이름은 그리스 신화에 등장하는 천하장사인 '헤라클레스'의 영문표기이다. 때문에 허클리스가 국내에 처음 들어왔을 무렵에는 허클리스를 '헤라클레스'라고 발음하는 사람들이 상당수 있었다. 그런데 'HERCULES' 의 영어 발음이 허클리스라는 것을 알고 부터는 모두가 허클리스로 발음하기 시작한 것이다. 그런데 오히려 지금은 허클리스는 알아도 이 단어의 우리말 발음이 헤라클레스라는 것을 모르는 사람들이 많아졌다. 비교적 짧은 기간에 말이 변해버린 한 예일 것이다.

그런데 이 카드의 이름이 하필이면 허클리스냐 하는데는 여러가지 추측이 있으나 가장 정통한 것은 다음과 같다. 허클리스 컴퓨터 테크놀리지사가 이 카드를 생산하기 시작할 무렵 이 카드를 팔기위한 광고 카피로 "We're strong on graphics"라는 말을 사용했고 힘센 거인의 대명사인 허클리스를 자사의 이름과 상품에 쓰게 되었다는 것이다.

허클리스 카드는 풀 사이즈의 카드로 이 카드에는 여러개의 IC가 장착되어 있다. 그중에서 핵을 이루는 것은 CRTC인 MC6845이고 다음에는 글자 모양을 담고 있는 문자롬이다. 또한 어드레스 디코딩용의 회로와 프린터쪽에 관계된 회로, 그리고 비디오 신호를 합성해내는 회로들로 구성되어 있다. 자세한 회로에 관한 것은 필자가 하드웨어 전문가가 아니기 때문에 생략하기로 하겠다. 원래 비디오에 관계된 회로는 무척이나 어렵다. 이는 타이밍 문제가 크기 때문이다.

아직도 세운상가에서 만들어지고 있는 한글카드(허클리스를 이용한)가 도트가 옆으로 밀리고 하이라이트시에 번지고하는 것은 이러한 부분의 타이밍 처리가 어렵고 고치기 힘든 탓임을 알아두시기를 바란다.

(1) 허클리스 카드의 특성과 제어 어드레스

허클리스는 IBM의 비디오램 128K 중에서 64K만을 사용하고 있다. 이 64K는 다시 32K씩의 두 뱅크로 나누어지고 이는 두개의 그래픽 페이지 버퍼로서 사용되고 있다. 프린터용 인터페이스는 시스템의 첫째번 프린팅디바이스로 사용된다(도스의 LPT1:을 뜻한다). 여기에는 대부분의 프린터가 접속될 수 있다.

① 디스플레이용 버퍼

페이지 0 : B000~B7FF(32K바이트)
페이지 1 : B800~BFFF(32K바이트)

② 입/출력 포트

(출) 03B4 : 6845 인덱스 레지스터
(입/출)* 03B5 : 6845 데이터 레지스터
(출) 03B8 : 디스플레이모드 제어포트
(출) 03B9 : 라이트펜 F/F 세트
(입) 03BA : 디스플레이 상태표시 포트
(출) 03BB : 라이트펜 F/F 리세트
(입/출) 03BC : 프린터 데이터 포트
(입) 03BD : 프린터 상태 포트
(입/출) 03BF : 프린터 제어 포트
(출)** 03BF : Configuration 스위치

③ 인터럽트 요청

IRQ 7…이 인터럽트는 프린터에서 액크놀리지 데이터가 올때 일어난다. 이것은 프린터 제어포트인 03BE가 사용중에는 masked off된다.

④ 논리적 장치의 이름

Display=CON:(모노크롬 디스플레이)
printer=LPT1:

(2) 허클리스의 제어

허클리스 그래픽카드는 아주 강력한 그래픽용 장비이다. 다만 이것을 제어하는데는 아주 섬세한 주의가 필요하다. 잘못 제어하게 되면 모니터에 손상을 입힐 수가 있다(사실은 아주 잦다). 필자도 비디오카드를 제어하는 프로그램을 여러번 작성하였는데 몇번인

* Non DMA only
** 모델번호 GB101은 Configuration 스위치를 소프트웨어로 제어할 때 페이지 1이 masked off된다. 뒤에 자세히 설명하겠다.

가 계산 실수로 프로그램을 잘못하여 모니터를 태워 먹을 일이 있다. 프로그래밍을 실수하거나 또는 프로그램이 폭주하였을때 나타나는 현상에 대하여 몇가지를 열거하겠다. 만일 이러한 현상이 나타나면 즉시 모니터의 전원을 꺼주기 바란다. 오랫동안 이러한 현상을 방관하다가 모니터가 고장나면 필자의 충고를 무시한 벌로 생각해 주기 바란다.

먼저 화면에 글자가 나오지 않고 이상하게 글자가 흐른다든지, 화면이 막 떨리고 모니터에서 소리가 난다든지, 혹은 화면의 정가운데 쯤에 굉장히 밝은 빛이 나타나면서 연기가 난다든지 할 경우는 모니터의 전원을 꺼야 한다.

① 디스플레이 부분의 인터페이싱

모노크롬 디스플레이 부분의 인터페이싱은 아래의 4개 입/출력 포트를 통해 일어난다.

03B4 : 6845 인덱스 레지스터
03B5 : 6845 데이터 레지스터
03B8 : 디스플레이모드 제어포트
03BA : 디스플레이 상태포트

㉠ 6845의 인덱스와 데이터 레지스터 (03B4/03B5)

6845는 모노크롬 디스플레이의 주파수와 스크린의 폭과 높이를 제어하며 16개의 파라미터가 제공된다. 제어는 먼저 03B4에 레지스터의 번호를 넣고 03B5에다 데이터를 넣거나 읽어오는 방법을 취하도록 설계되어 있다. 만일 입/출력 어드레스에 여유가 있으면 16개의 레지스터를 모두 각각의 번지에 지정할 수도 있었을 것이나 IBM-PC에는 그럴만한 여유가 없다.

대개 이 레지스터의 값을 조정해서 프로그래밍 할 경우에 절대로 고급언어를 쓰지 말 것을 당부한다. 고급언어의 경우 섬세한 제어가 어렵고 또한 외부요인에 의해 방해를 받기 때문에 위에서 말한 피해를 볼 경우가 많기 때문이다. 필히 어셈블러를 사용하기 바란다. 프로그래밍법은 아래와 같은 방식을 취한다.

```
MOV AX,03B4H
MOV DX,AX
MOV AL,레지스터 번호
OUT DX,AL            :레지스터의 지정
MOV AL,데이터 값
INC DX
IN or OUT DX,AL      :데이터의 입/출력
```

일단 6845의 레지스터와 그 쓰임새에 관하여 먼저 알아보고나서 거기에 어떤 데이터를 어떤 계산공식에 의해서 넣는지 설명하겠다(표 1 참조).

㉡ 디스플레이 모드 제어포트(03B8)

이 출력포트는 그래픽 카드의 모드를 조정한다.

0 : 사용되지 않는다.
1 : 0=텍스트 모드(전원투입시 설정치)
  1=그래픽 모드(6845는 이 값이 바뀔때마다 다시 프로그래밍을 해 넣어야 한다.)
2 : 사용되지 않는다.
3 : 0=화면이 꺼진다.(전원투입시 설정치)
  1=화면이 보인다.(이 비트는 화면을 그래픽이나 텍스트에서 다른 상태로 바뀔때 유용하게 쓰인다. 이를 끄고 모드를 바꾸면 화면 절환시의 번쩍거림이 사라진다.)
4 : 사용되지 않는다.
5 : 0=글자가 껌벅거리는 것을 못하게 한다(전원투입시 설정치)
  1=글자를 껌벅거릴 수 있게 한다. (이것은 커서에는 전혀 영향을 미치지 않는다. 각 글자는 자신의 어트리뷰트를 감지하고 껌뻑거린다.)
6 : 사용되지 않는다.
7 : 0=페이지 0(전원투입시 설정치, B0000 부터 시작한다.)
  1=페이지 1(B8000부터 시작한다. 이 비트는 그래픽 모드에서만 작동한다.)

디스플레이 모드 제어포트를 건드릴때는 다음의 사항들을 주의해야 한다. 먼저, 텍스트와 그래픽모드를 바꿀때는 아주 주의가 필요하다. 잘못하면 심각한 사태에 빠지게 된다.

그리고 텍스트 모드와 그래픽모드를 바꿀때 화면에 지정하지 않은 수평과 수직 주파수가 투입되게 된다. 이럴때 고급언어를 사용하여 이 값을 바꾸려 들지 말아야 한다.

㉢ 디스플레이 상태 포트

이 입력 포트는 모노크롬 디스플레이의 실시간 감지에 사용된다.

0 : 0=정상글자
  1=SYNC(화면이 일시적으로 없어진다.)
1 : 사용되지 않는다.
2 : 사용되지 않는다.
3 : 0=Dots off
  1=Dots on(모노크롬 디스플레이가 Active 비디오 신호를 받고 있는지를 소프트웨어처럼 검사위치 설정, 검증 하는데 쓰인다.)
4 : 사용되지 않는다.
5 : 사용되지 않는다.
6 : 사용되지 않는다.
7 : 0=수직 retrace(화면이 일시적으로 안보인다.)
  1=active 디스플레이 (이 정보는 소프트웨어가 화면을 비게하는데 유용하다.)

② 프린터 부분의 입/출력 인터페이싱

프린터 부분의 입출력은 아래의 3 포트에 의해서 이루어진다.

03BC : 프린터 데이터 포트
03BD : 프린터 상태 포트
03BE : 프린터 제어 포트

㉠ 프린터 데이터 포트(03BC)

이 포트에로의 출력은 데이터 버스로부터 값을 프린터에 래치시킨다. Actual Pin상태는 입력시 되돌려 진다.

㉡ 프린터 상태 포트(03BD)

이 포트는 단지 입력만이 가능하다.

0 : 사용되지 않는다.
1 : 사용되지 않는다.
2 : 사용되지 않는다.
3 : 0=프린터 에러
  1=정상 동작
4 : 0=프린터가 온라인이 되지 않았다.
  1=프린터가 온라인 상태에 있다.

〔표 1〕 6845 레지스터의 용도

인덱스	데 이 터
0	SYNC를 포함하여 한줄에 나타날 글자의 총 수
1	한줄에 표시되어 보일 글자 수(이것이 눈에 보이는 글자 수다.)
2	SYNC동안에 표시될 글자의 첫번째 위치, 1보다 작아야 한다.
3	각 줄의 SYNC동안의 글자 수, 1보다 작아야 한다.
4	줄의 수로 1보다 작아야 한다. 수직 retrace동안의 줄을 포함한다.
5	scan의 수로 전체 줄의 수를 더한 값이다.
6	볼 수 있는 줄의 수
7	retrace를 시작하는 줄의 값, 1보다 작아야 하며, 항상 마지막 16scan이다.
8	항상 0이 출력된다(이것은 인터레이스와 skew인데 6845 데이터 시트를 참고하라.).
9	각 줄의 scan 수, 1보다 작아야 한다.
A	커서가 글자에 겹치는 출발 위치
B	커서가 글자에 겹치는 마지막 위치(Ins 키를 눌렀을 때 커서가 바뀌는 것을 본 적이 있을 것이다. 이 레지스터를 조절하고 있다.)
C	비디오램의 시작 레지스터이다. 자세한 것은 데이터 북을 참조
D	위의 것은 상위 값이고, 이것은 하위 값이다.
E	커서가 위치할 메모리 번지, 상위 값
F	위의 하위 값
10	버퍼안의 오프셋, 라이트펜에서 검지시의 상위 값
11	위의 하위 값

① 모노크롬 디스플레이는 대략 54μs의 스캔(scan)이 필요하다.
② 전체 줄의 수와 스캔의 수는 필히 화면이 초당 50번 업데이트될 충분한 시간으로 조정되어야 한다.
③ 그래픽카드의 local oscillator는 텍스트 상태에서 0.5625μs의 시간 기저를 생산한다. 그리고 bit-mapped상태에서는 각 글자당 1μs가 소요된다.
④ Bit-mapped상태에서 한 글자는 16도트의 폭과 4스캔의 높이를 갖는다. 텍스트 모드에서는 9도트의 폭과 14스캔의 높이를 갖는다. 따라서 초기의 세운상가 한글카드의 한글은 14스캔안에서 설계되었기 때문에 글자가 매우 흉하다.

5:0=정상 동작
1=종이가 없다.
6:0=데이터를 받고 있다.
1=정상 동작
7:0=프린터가 Busy상태에 있다.
1=프린터가 Busy상태가 아니다.

㉢ 프린터 제어 포트(03BE)

이 포트는 입/출력이 모두 가능하다.

0:0=프린터의 스트로브가 데이터 래치로부터 데이터를 받는다.
1=스트로브를 릴리즈한다.(전원투입시 설정치)
1:0=자동 행바꿈
1=행바꿈을 제어해야 한다.(전원투입시 설정치)
2:0=프린터의 초기화.(전원투입시 설정치)
1=정상상태로 동작한다.
3:0=프린터를 선택하지 않는다.
1=프린터를 선택한다.(전원투입시 설정치)
4:0=IRQ 7을 들어오지 못하게 한다.(전원투입시 설정치)
1=IRQ 7을 프린터가 데이터를 받을 준비가 되면 가능케 한다.
5:사용되지 않는다.
6:사용되지 않는다.
7:사용되지 않는다.

프린터의 제어에 관한 방법은 상당히 많이 소개되어 있으니 이 글에서는 이에 관한 것을 생략하기로 하겠다.

㉣ Configuration 스위치(03BF)

이 포트는 그래픽카드의 세가지 메모리 매핑과 그래픽모드 세팅의 사고를 막기위해서 있다. 세팅법은 앞에서 이미 설명하였으므로 생략하기로 한다.

허클리스 카드의 메모리 구조

허클리스 카드 역시 정상적인 비디오 디스플레이/프린터이므로 특별히 다른 카드와 다른 것은 없다.

다만 화면상의 어드레스 배열과 지정법이 다르므로 이에 관하여 알아 보기로 하자.

(1) 화면상의 글자의 위치를 계산하는법(TEXT)

텍스트 화면은 B0000–B0FFF까지 4K바이트의 영역을 가지고 있으며 논리적인 문자의 코드와 그 속성(Attributes) 을 메모리에 넣으면 화면에 그 글자가 나타나는 방식을 취하고 있다. 따라서 메모리의 위치가 상대적인 화면상의 글자의 위치에 대응된다. 이를 계산하는 방법에 대해서 알아보기로 하자.

텍스트 디스플레이는 가로 80자, 세로 25줄로 구성된다. 모든 글자는 한줄에 160바이트씩의 연속적인 메모리에 저장된다. 각 글자는 한개의 코드 바이트와 하나의 특성 바이트를 갖는다.

글자의 코드가 저장되는 오프셋은 아래와 같이 계산한다.

```
= 160*(LINE-1)+2*(COLUMN-1)
```

글자의 특성이 기억되는 오프셋은 또 다음과 같이 계산된다.

```
= 160*(LINE-1)+2*(COLUMN-1)+1
```

여기에서 LINE은 1부터 25사이의 숫자이고 COLUMN은 1부터 80사이의 값을 갖는다.

따라서 여기에서 구한 오프셋에 코드나 특성 코드를 넣으면 해당 위치에 글자나 특성이 바뀌게 된다. 화면을 '@'자로 가득채우는 베이직 프로그램을 간단히 작성해보면 다음과 같다.

```
10 CLS                      :화면을 지움.
20 DEF SEG=&HB000           :비디오램 세그먼트지정
30 FOR I=0 TO 8191 STEP 2:글자,특성코드첨가,+2
40  POKE I,ASC('@')         :'@'를 오프셋에 넣음
50  POKE I+1,7              :정상글자지정 특성코드
60 NEXT I
70 END
```

허클리스는 하드웨어적인 글자 생성기로 256개의 다른 글자들을 그 글자

의 코드로서 생성해내며 어트리뷰트 디코더는 밑줄, 역상, 빈칸, 굵은 글자, 깜빡임등을 생성해 낸다. 글자 생성기는 표준 IBM-PC의 글자형과 같은 글자체를 제공한다. 이들 특성은 서로 조합이 가능하다.

이들 특성 디코더는 다음과 같은 방식을 가지고 있다.

7 6 5 4 3 2 1 0	Attribute Codes
B 0 0 0 I 0 0 0	Blank
B 0 0 0 I 0 0 1	Underline
B 0 0 0 I 1 1 1	Normal Display
B 1 1 1 I 0 0 0	Reverse Video

여기에서 I=0 정상굵기

I=1 굵은 글씨

만일 깜빡임을 지웠을 때(디스플레이 모드 제어 포트의 비트 5=0)

B=0 정상적인 글자의 배경

B=1 굵은 배경

또한 깜빡임을 켰을 때는 (디스플레이 모드 제어포트의 비트 5=1)

B=0 깜빡이지 않음

B=1 깜빡임

(2) 화면상의 점의 위치를 계산 하는법(그래픽)

허클리스가 그래픽 상태에 있을때 디스플레이용의 버퍼는 두개의 페이지를 갖는다. 이 두개의 페이지는 교대로 표시된다. 즉 한번에 두페이지가 동시에 표시되지는 않는다.

페이지의 선택은 디스플레이 모드 제어포트의 비트 7을 아래와 같이 지정함으로써 가능하다.

0이면 페이지 0

1이면 페이지 1

버퍼의 영역은 아래와 같이 할당되어 있다.

페이지 0 : B0000—B7FFF

페이지 1 : B8000—BFFFF

각 페이지에서의 점(x,y)이 나타내는 위치(오프셋)는 아래와 같은 방법으로 구할 수 있다.

```
= [2000H * (Y MOD 4)] + [90 * INTEGER (Y/4)] +
  [INTEGER (X/8)]
```

그리고 도트가 그 바이트에서 몇번째 비트인가 하는 것은 아래와 같은 방식으로 계산되어 진다.

```
= 7 - (X MOD 8)
```

여기에서 X는 0부터 719 사이의 숫자이고, Y는 0부터 347 사이의 값을 갖는다.

예) 좌표(300, 250)의 오프셋

```
= [2000H * (250 MOD 4)] + [90 * INTEGER (250/4)]
  + [INTEGER(300/8)]
=[2000H * (2)] + [90  * (62)] + (37)
=4000H + 5617
=4000H + 15F1H
=55F1H
```

그리고 비트의 위치는

```
= 7 - (300 MOV 8) =7 - (4) =3.
```

지금까지 설명한 것은 허클리스가 720×348의 비트맵 형식을 가지고 있다는 전제하에 설명한 것이다. 만일 6845의 레지스터 조작으로 이를 640×400(NKP등이 이용하고 있음)이나, 768×341 (MHP등이 이용하고 있음)등으로 비트맵을 바꾸면 그에 따라서 계산하는 방법이 달라진다. 이때는 위의 공식중에서 '90'으로 나타낸 숫자를 바꾸면 되는데 이것이 의미 하는바는 다음과 같다.

90=720/8는 결국 수평 해상도를 8로(한 바이트의 비트수) 나눈 결과인 것이다. 따라서 640×400인 경우에는 80을 768×341인 경우에는 96을 90대신에 사용하면 된다. 이를 계산하는 어셈블리 프로그램을 작성할 때는 될 수 있으면 'DIV'나 'MUL'등의 명령어를 쓰지 말고 시프트를 이용하여 계산하기 바란다. 나눗셈이나 곱셈연산은 오랜 시간을 요하게 되며 어드레스 계산 루틴은 그래픽 프로그램에서 가장 자주 쓰이게 되는 부분이므로 이를 잘 짜야만 전체적인 프로그램의 속도를 높일 수가 있는 것이다. 예를 들어 '한컴퓨터 연구소'의 작품인 '한글워드'의 경우에는 이의 계산을 빨리하기 위하여 각 글자를 화면에 그릴 때 홀수열과 짝수열로 나뉘는 CGA의 특성을 고려하여 한줄씩 건너뛰어 한글자를 짝수열과 홀수열로 두번 그리는 방식을 취하고 있다.

여기에 허클리스의 그래픽 메모리 매핑을 표 2에 소개한다. 단 이것은 720×384용이며 페이지 0이다. 페이지 1의 매핑은 여기에 오프셋 8000H를 더한 값이다.

표 2의 매핑표를 보면 어떤 규칙성을 볼 수 있을 것이다. 네줄씩이 2000H 만큼씩의 차이를 보이며 연속적으로 되어 있다. 이런 매핑구조를 가지고 있기에 위의 공식이 성립하는 것이다. 즉 이런 구조를 modulo-4 구조라고 부른다. 또한 CGA의 매핑이 한줄 건너 한줄, 즉 modulo-2라는 것은 이미 앞에서도 언급한 적이 있다. 이 얘기를 들은 독자는 무언가 떠오르는 것이 있을 것이다. 이 생각이 떠오르지 않는 독자는 아마 게임을 한번도 해보지 않은 무지막지(?)한 독자일 것이다. 이것이 무엇이냐 하면 그 왜 SIMCGA라는 프로그램을 띄우고 게임을 하면 중간에 선이 두줄씩 빠지는 것을 본 일이 있을 것이다. 이제야 그 이유를 아신 분도 있을 것이다. 이것을 채우기 위해 일정시간에 한번씩 위의 줄을 아래로 복사해 넣는데 그 시간차가 벌어지기 때문에 화면이 흐르는 것처럼 보이는 것이다. 결국 해결책은 매직키(MKFILE or MKBOOT)와 같은 방식을 쓰는 것이다. 그 방법은 비교적 골치(?)를 썩히는 것이니 나중에 소개하기로 하겠다.

(3) 레지스터의 요약

지금까지 위에서 설명한 내용을 간단히 정리하여 도표로 작성하면 다음과 같이 할 수 있다.

허큘리스 카드에의 프로그래밍

허큘리스 카드를 이용한 여러가지 프로그램을 소개하였다. 이중 처음 것은 허큘리스 카드의 모드를 그래픽과 텍스트로 서로 전환할 때 이용되는 루틴으로 어셈블리 프로그램으로 작성된 것이다.

Code segments to change modes. (Make sure that the configuration bit is set to your requirement before attempting to change modes.)

```
;           port address
index       equ         03b4h
cntrl       equ         03b8h
;           control codes
scrn_on     equ         8
grph        equ         2
text        equ         20h

xdata segment public 'data'
gtable      db          35h,2dh,2eh,07h
            db          5bh,02h,57h,57h
            db          02h,03h,00h,00h
ttable      db          61h,50h,52h,0fh
            db          19h,06h,19h,19h,
            db          02h,0dh,0bh,0ch
xdata ends
xcode segment public 'code'
            assume cs:xcode,ds:xdata
```

GRAPHICS MODE : 6845 CRT 콘트롤러를 720×348의 그래픽 모드로 프로그래밍한다. 화면에 보여지며 활용하는 페이지는 페이지 0이다.

```
gmode       proc        near
            push        es
            push        ds
            mov         ax,data
            mov         ds,ax
            mov         al,grph
            lea         si,gtable
            mov         bx,0
            mov         cx,4000h
            call        setmd
            pop         ds
            pop         es
            ret
```

Bit#	CRT Control (03B8) (Write Only)	CRT Status (03BA) (Read Only)	Printer Data (03BC) (Read/Write)	Printer Status (03BD) (Read Only)	Control (Read/Write)
0	Not used	+Horizontal Sync	D0	Not used	+Strobe
1	+Select Graphic Mode	Reserved	D1	Not used	+Auto Feed
2	Not used	Reserved	D2	Not used	-Initialize
3	+Enable Video Output	+Video Output	D3	-Error	Printer
4	Not used	Not used	D4	+Select Status	+Select In
5	+Enable Char Blink	Not used	D5	+Paper Out	+Enable IRQ7 On -ACK
6	Not used	Not used	D6	-ACK	Not used
7	Page #	-VSync	D7	-Busy	Not used

```
gmode       endp
```

TEXT MODE : 6845를 프로그래밍하고 CRT 콘트롤레지스터를 텍스트모드로 바꾼다.

```
tmode       proc        near
            push        es
            push        ds
            mov         ax,xdata
            mov         ds,ax
            mov         al,text
            lea         si,ttable
            mov         bx,720h
            mov         cx,2000
            call        setmd
            pop         ds
            pop         es
            ret
tmode       endp
setmd       proc        near
;   sets mode는 그래픽이나 텍스트를
;   al값에 의해서 결정한다.
;   si =파라미터 테이블
;   cx =clear할 워드수
;   bx =블랭크의 값
            push        ds
            push        es
            push        ax
            push        bx
            push        cx
;   scrn_on 없이 화면을 바꾼다.
            mov         dx,cntrl
            out         dx,al
;   6845를 초기화 한다.
            mov         ax,ds
            mov         es,ax       ;also point es:sdi
                                    ;to parameter
                                    ;table
            move        dx,index
            mov         cx,12       ;12 parameters to
                                    ;be output
                                    ;starting from
                                    ;reg,0
parms:      mov         al,ah
            out         dx,al       ;output register
                                    ;number
            inc         dx
```

〔표 2〕 허클레스의 그래픽 메모리 매핑

Y \ X	0	8	16	24	....	472
LINE 0	0000	0001	0002	0003	....	0059
LINE 1	2000	2001	2002	2003	....	2059
LINE 2	4000	4001	4002	4003	....	4059
LINE 3	6000	6001	6002	6003	....	6059
LINE 4	005A	005B	005C	005D	....	00B3
LINE 5	205A	205B	205C	205D	....	20B3
LINE 6	405A	405B	405C	405D	....	40B3
⋮	⋮	⋮	⋮	⋮	....	⋮
LINE 347	7E3C	7E3D	7E3E	7E3F	....	7E95

```
        lodsb
        out     dx,al       ;output data
        inc     ah          ;next value
        dec     dx
        loop    params
        pop     cx          ;clear the buffer
        mov     ax,0b00h
        cld
        mov     es,ax
        xor     di,di
        pop     ax
        rep     stosw
        ;  scrn_on,page0
                mov     dx,cntrl
                pop     ax
                add     al,scrn_on
                out     dx,al
                pop     es
                pop     ds
                ret
setmd           endp
xcode ends
```

위의 프로그램을 독자의 어셈블리 프로그램에 삽입하여 사용하거나 따로 어셈블해서 다른 고급언어에서 불러 쓰면 될 것이다. 위의 프로그램중 'g-table' 의 값만을 바꿈으로 해서 가로 도트수와 세로 라인수가 바뀌게 된다. 다음에 그 예를 들어 설명하겠다.

```
gtable  db      31h,28h,29h,08h
        db      68h,02h,64h,65h
        db      02h,03h,02h.01h
```

위의 gtable은 640×400의 매핑을 가지는 파라미터 값이다. 이 값을 계산해 내는 것은 무척이나 어려운 작업이다. 어쨌든 위와 같이 gtable을 바꾸어 위의 프로그램에 넣으면 640×400의 그래픽 화면을 만들 수 있다. 이것은 NKP등의 프로그램이 이용하고 있는 방법이다. 이 파라미터를 만들기 위해서는 상당히 어려운 공식과 하드웨어의 전 스펙을 알아야하는 난제가 따른다. 이것을 case by case로는 알아낼 수가 없으며 또한 이렇게 하다가는 모니터가 견디지 못할 주파수가 발생하여 모니터가 고장나는 불운을 당하게 될지도 모른다. 하여간에 주의를 하여 주기 바라며, 비트맵의 포맷을 바꾸는 공식은 모토롤라사 발행의 데이터 북을 참고하여야 할 것이다.

지면 관계상 그 내용을 게재하지는 못하겠다. 인터레이스 모드에 관한 내용 역시 데이터북에 자세히 설명되어 있으므로 여기서는 언급을 회피하기로 하겠다.

끝마치면서

별 내용도 없는 글(?)을 끝까지 읽어주신 독자 여러분께 우선 감사를 드린다. 좀더 자세한 내용들을 다루려 했으나 필자의 시간이 그에 따르지 못함을 아쉽게 생각한다. 하드웨어를 설계하고 또 그를 제어하는 프로그램을 작성하는 것은 대단히 어렵고 또 고통스러운 일이다. 필자가 시간이 나면 허클리스 카드를 제어하는 여러가지 유틸리티들을 발표할 것을 약속드린다.

시중에 허클리스를 이용하여 그래픽을 제어하는 유틸리티로 GRAPHX라는 프로그램이 있다. 이 프로그램을 이용하면 아쉬우나마 제한된대로 그래픽을 할 수가 있다. 또 한 여러개의 프로그래밍 언어에서 허클리스를 지원하고는 있다. 하지만 궁극적으로 EGA급의 디스플레이 카드가 표준으로 되어 가고 있고 사람들 역시 좀더 많은 컬러와 해상도를 요구하고 있어 오래지 않아 허클리스 역시 사양길에 접어들 것이다. 허클리스사의 다음 제품인 허클리스카드 플러스와 허클리스 컬러카드에 대한 리뷰는 자료가 도착하는대로 써 볼 것을 다짐한다. 이글이 독자여러분께서 허클리스를 이해하는데 조금이나마 도움이 되기를 바라며 글을 마친다.

아래에 게재된 프로그램은 일명 '지진'이라는 프로그램인데 실행시켜 보면 왜 그런 이름을 붙였는지 이해할 수가 있을 것이다. PC

```
10 REM SHAKE AND SHIFT SCREEN
20 DEF SEG = &HB000
30 '
40 OUT &H3B4,13
50 FOR I = 0 TO 2000
60 OUT &H3B5, RND(1) * 255
70 NEXT I
80 OUT &H3B5,0
90 '
100 FOR I = 2047 TO 0 STEP -1
110 HI = I  256
120 LO = I - HI * 256
120 OUT &H3B4,12
140 OUT &H3B5, HI
150 OUT &H3B4, 13
160 OUT &H3B5, LO
170 NEXT I
```

국내소식

–과기처

컴퓨터등 35종 국산화 기술고시

과기처는 지난 1월 6일 데이터베이스 컴퓨터 등 35종의 기계류 부품, 소재를 국산화 개발대상 기술로 고시했다.

과기처는 기술고시품목에 대해 연구개발, 기술지도, 기술자문훈련 등 기술지원과 금융, 세제, 신기술보호 등 각종 지원수단을 통해 국산화 개발을 촉진할 계획이다.

이번에 고시된 기술품목은 다음과 같다.

- 최적지속파 신호처리시스템
- 셀프튜닝 콘트롤러
- 셀룰러폰용 송수신분리형 세라믹여파기
- 열량측정용 방사선회분계
- 급속열처리장비
- 보일러 집진탈황을 겸한 폐열회수기
- 산업폐기물 소각장치
- 석탄처리공정의 수분자동측정 및 제어시스템
- 중앙집중갱내 감시장치
- 소형정밀품 생산라인의 외관검사기
- 스캐너를 이용한 문서인식, 처리기
- 데이터베이스 컴퓨터
- 영상전화용 영상부호화기
- 자동차 엔진제어용 IC
- 음성다중 TV용 투캐리어 IC
- 랩탑PC 디스플레이용 IC
- 비디오 디스플레이 제어 IC
- 중형LCD TV용 커스텀 IC
- 갱내 구명구
- 마이크로프로세서를 이용한 전자식 배전반
- EVA 에멀전 공중합기술
- EVOH 펠레트 공중합기술
- 아크릴수지 중합기술
- 자외선 경화수지
- 자외선 경화형 솔더 레지스터 중합 및 성형
- 자동차 연료탱크 도금
- 반도체칩 증착재
- 내열 내마멸 코발트합금
- 고체윤활제 성형가공
- 절전형램프 형광안료합성
- 부탄을 이용한 새성형탄 제조
- 탄소섬유를 이용한 시멘트복합재와 응용제품
- 의료용 접착제

–한국정보산업연합회

기술자문위원회 설치키로

한국정보산업연합회(회장 : 이용태)는 소프트웨어 복제사건을 과학적이고 합리적인 방법으로 해결키 위해 복제사건이 법정 싸움으로 비화하기 이전에 쌍방의견을 조정할 수 있는 「컴퓨터프로그램 저작권 분쟁조정위원회」와 소프트웨어 불법복제, 개작 등 기술적 감정 및 자문을 전담할 「기술자문위원회」를 설치 운영키로 하고 과기처와 협의중에 있다.

법조계, 학·연구계, 업계, 정부 관련기관 20~30명의 전문가로 구성될 저작권 분쟁조정위원회에서는 프로그램복제 사건의 이해, 조정 이외에도 ●컴퓨터 프로그램 보호법의 합리적인 운영 ●프로그램 이용환경개선 ●프로그램저작권 침해에 대한 구제방안 제시 ●소프트웨어 상품화 인식제고 등을 전담한다.

또한 정보처리기술사 등 정부 및 업계·소프트웨어엔지니어 50여명으로 구성될 기술자문위원회는 소프트웨어 저작권 침해여부에 대한 기술적 감정 및 자문을 비롯, 법해석상의 기술적 의견자문·정보처리기술력 제고·프로그램 유통촉진 업무를 담당할 예정이다.

–한국전자통신연구소

경산 등 4개지역 TDX-1B 개통 예정

국산 전전자식 교환기 TDX–1A의 성능과 용량을 배가시킨 TDX–1B가 개발돼 농어촌 및 중소도시의 통신망 현대화는 물론 고도 정보화사회의 근간이 될 종합 정보통신망 구축을 위한 대용량의 전전자교환기 TDX–10의 조기실용화가 가능해졌다.

한국전자통신연구소 주관으로 금성반도체·대우통신·동양전자통신·삼성전자·동아전기 등 5개 업체가 지난 86년6월 공동개발에 착수, 2년6개월만에 개발완료된 이 TDX–1B는 지난 1월10일 전자통신연구소에서 가진 발표회를 통해 알려졌다.

이 TDX–1B는 농어촌용으로 주로 사용되고 있는 1만회선 용량의 TDX–1A의 개발기술을 활용, 최소한의 기술변경으로 개발된 2만3천회선의 중소도시용 교환기로 그동안 서대전 전신전화국에 모델기를 설치해 시험서비스

를 실시한 결과, 양호한 것으로 밝혀졌다.

이에 따라 한국전기통신공사는 오는 4월부터 경산·주문진 등 4개지역에 TDX-1B를 시범개통할 계획이다.

–한국정보과학회

제7차 전산학교수세미나 개최

한국정보과학회(회장 : 이철희)는 지난 1월20, 21일 양일간 인천 송도비치호텔에서 제7차 전문대학 전산과교수 세미나를 개최했다.

이번 세미나에서는 이태원 고려대교수가 「순서논리회로의 해석과 설계」, 이기준 서울대교수가 「대학 교육전산망 구축을 위한 기본계획」 등의 주제를 발표하였다.

–한국전기통신공사

고성능 단말기술 개발 계획

한국전기통신공사는 전화망을 이용한 정보통신서비스가 확대될 것에 대비하여 올해부터 한국전자통신연구소(ETRI)와 학계 및 금성반도체, 삼성전자 등의 관련업체와 공동으로 고속복합단말기와 광대역 지능형 정보통신 단말 기술을 개발키로 했다.

따라서 이에 대한 개발계획을 올해부터 3단계로 나누어 본격적으로 추진할 계획이다.

1단계는 89~91년으로 텔렉스에 워드프로세서 기능을 추가한 텔리텍스를 비롯한 문자형 단말 기술과 G4급 고속 팩시밀리, 화상전화 등의 기술개발에 주력할 계획이다.

2단계는 92~2천1년으로 잡고 매체간 변환기술, 문자와 화상복합 단말기술, 음성 및 화상압축 기술, 음성과 문자정보 입출력 기술 등 복합 광대역 서비스기술을 중점 개발할 계획이다.

3단계는 2천2~2천11년으로 고기능 서비스단말 기술인 인공지능형 종합워크스테이션과 화상정보 입출력 기술개발에 나설 방침이다.

–한국전기통신공사

전송로 신설용으로 폼스킨케이블 발주

한국전기통신공사는 통신선로 현대화를 위해 폼스킨케이블을 금성전선, 대한전선, 국제전선, 일진, 대성전선 등으로부터 대량 구매키로 함에 따라 우리나라도 올부터 본격적인 폼스킨시대로 접어들게 되었다.

폼스킨케이블의 공급 첫해를 맞이하여 통신공사는 2백 45억6천만원(1천5백km) 상당의 입찰을 실시했는데, 이번 입찰에서는 금성전선이 1백회선 등 20종 87억1천7백만원어치를, 대한전선이 2백회선 등 22종 87억3백만원어치를, 국제전선이 3백회선 18종 36억8천만원어치를 각각 수주하여 전체의 90%(2백11억원)에 해당하는 점유율을 보였다.

이들 전선 3사외에 일진은 3백회선 2종에 대해 34억2천만원어치를 수주했으며, 대성전선이 25회선 등 6종으로 3천7백만원을 수주하여 중소전선업체로서는 처음으로 통신공사의 폼스킨 공급업체가 됐다.

이번 입찰에서 수주를 따낸 메이커들은 폼스킨케이블을 이달부터 분할공급하게 되는데, 통신공사는 이 폼스킨케이블을 전국 86개 전화국의 기간전송로 신설용과 서울지역의 낡은 종이절연케이블 대체용으로 사용할 방침.

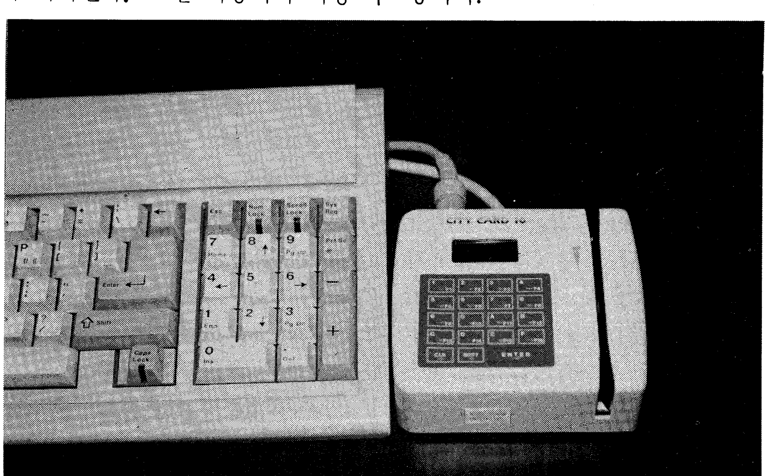
–경기시스템(주)

CITY CARD-10K 개발

경기시스템(주) (대표 : 김승찬)는 신용카드 조회용 단말기인 CITY CARD-10의 새 버전 CITY CARD-10K를 개발 완료했다.

CITY CARD-10K는 기존의 PC나 CRT, 터미널 등 단말기의 키보드에 직접 붙여서 사용할 수 있는 카드리더로서 통신포트와 전원공급등을 추가로 설치할 필요가 없어 경비절감의 효과가 기대된다. 또한 사용자가 사용하는 단말기의 사양에 따른 주문 생산도 가능하다.

동 제품의 특징은 카드리더를 통하여 한번에 데이터를 보내므로 키보드를 두드릴 때의 에러를 방지할 수 있으며 키보드와 카드리더가 데이터를 공유할 수 있다는 점 등을 들 수 있다. 활용범위로는 은행카드나 증권카드 또는 병원카드, 항공사의 보너스카드를 이용하는 곳 이외에도 마그네틱 스트라이프 카드를 사용하는 곳은 모두 가능하다.



–금성계전

보급형 인버터 개발

금성계전(대표 : 백중영)은 가변속모터를 대체할 수 있는 소형, 염가, 보급형의 인버터 '스타버트' 시리즈를 개발, 보급에 나섰다.

금성계전이 청주공장에 양산체제를 갖추고 생산에 들어간 이 인버터는 기존 제품보다 크기가 3분의 1정도의 소형이면서도 가격이 10만원대로 저렴하여 운반, 하역, 기계, 섬유, 식품공업 및 컨베이어의 모터 속도제어와 자동화에 널리 사용될 것으로 기대된다.

스타버트 시리즈 인버터는 운전시 소음이 발생치 않는 특징이 있어 그동안 효율이 낮고 자동제어가 원활하지 못했던 가변속모터를 사용하던 생산설비업체를 중심으로 수요가 크게 늘어날 것으로 보인다.

–금성반도체

모뎀, 품질관리 1 등급 자격 획득

금성반도체(대표 : 이만용)는 모뎀으로 국내에서 최초로 공업진흥청으로부터 품질관리1등급 자격을 획득했다.

공업진흥청이 지정하는 품질관리 1등급 품목이 되려면 서류심사 및 제품검사, 생산라인의 실사 등 170여가지의 까다로운 심사기준을 통과해야 하는데, 품질관리 1등급 품목으로 지정되면 공진청이 고시한 품질검사 지정상품이라도 품질검사를 면제받게 되고, 전기용품 안전관리법에 따른 사후관리때 별도로 공장검사를 받지 않아도 되며, KS표시 허가신청때 필요한 공장심사도 면제된다.

이외에 수출검사를 면제받고 계량기구조에 대한 검증도 필요없으며, 수출품에 대한 세관검사, 국가기관 소요물량의 우선구매 등 특전이 주어진다.

공업진흥청은 81년부터 국내 제조 및 가공업체를 중심으로 공장품질관리 등급제를 운영하고 있는데, 이는 공산품의 품질을 높이고 공공의 이익과 소비자를 위한 것으로, 공진청의 품질관리 1등급 자격획득은 기술과 제품품질의 신뢰성을 인정받는 척도가 된다.

–금성산전

조명제어시스템 개발

금성산전연구소(소장 : 최호현)는 조명을 효과적으로 관리함으로써 에너지 절감을 도모할 수 있는 조명관리 자동제어시스템을 개발했다.

금성산전연구소 전력전자연구실이 지난 87년말부터 약3억원의 연구개발비를 투자하여 개발에 성공한 조명제어시스템은 한쌍의 전용신호선으로 다수의 조명기구를 개별 또는 조합하여 제어하는 시스템으로 마이크로컴퓨터를 내장한 주조작반과 조명기구를 제어하는 단말기로 구성되는데, 1대의 단말기로 최대 16회로의 부하를 제어하여 총4천회로의 부하를 주조작반에서 제어, 감시할 수 있다. 또한 개별점멸과 24종류의 조명패턴을 기억시켜 재현이 가능한 패턴점멸, 태양광이 충분할 때 자동적으로 조명을 일시 소등하는 센서점멸이 가능하며, 스케줄에 따른 자동제어 등 주조작반의 동작표시 LED에 의해 전체 조명기구의 점멸상태 및 주변기기의 동작상태를 집중감시할 수 있다.

–금성하니웰

디지탈 자동제어 콘트롤러 개발

금성하니웰(대표 : 홍성재)은 마이크로프로세서를 채택하여 온도나 습도, 압력, 유량, 회전속도 등을 프로그램에 의해서 자동제어할 수 있는 디지탈 자동제어 콘트롤러를 개발했다.

이 자동제어 콘트롤러는 설정된 데이터가 시간이 흐름에 따라 변경되도록 프로그램을 입력하면 순차적으로 제어가 진행되는 프로그램과 맵(map) 방식을 채용하고 있다.

또한 18종의 데이터를 동시에 입력할 수 있고 AC90~264V 전원에 별도의 전압조정없이 사용이 가능한 프리(Free) 볼트방식을 채택하고 있다.

이는 주로 연구소와 전기로, 건조로, 열처리로, 범용 항온조, 식품기계와 가공, 약품, 세라믹, 콘크리트, 자동차 등의 제조공정 자동화에 적합하다.

–경희대학교

정보관련 3개학과 신설

경희대학교 산업정보대학원은 정보화 사회로의 본격적인 진입에 대비하여 정보통신관련 전문기술자 양성과 기존의 기술인력 재교육을 위해 89년도부터 정보통신학과와 계측제어학과, 경영정보학과를 신설키로 하였다.

국내에서는 처음으로 대학원과정에 정보통신 관련학과를 신설하는 것으로 정보통신학과는 위성통신, 데이터 전송, 이동체 통신 시스템 설계와 통신망 계획, 통신 정책론 등 정보통신 관련 17개 과목을 개설하여 시스템 엔지니어를 양성하게 되며 경영정보학과는 16개 과목을 개설할 예정이다.

– 웅진출판

CBE사업 진출

웅진출판(대표 : 윤석금)은 일부 대학교나 연구기관에서 부분적으로 실시하였던 CBE(컴퓨터이용 교육)가 최근 컴퓨터 수요가 확대됨에 따라 초·중·고등학교에 확산되자 이 사업에 본격적으로 진출하였다.

동사는 초·중·고등학교의 영어·수학·과학교육 소프트웨어를 개발하여 데이콤 네트를 통해 터미널을 갖춘 학생들이 손쉽게 개별 학습을 할 수 있도록 한국 데이타통신의 통신망이용 계약과 미국의 와이켓 시스템등 CBE 장비도입을 추진하고 있다. 또한 편집 및 전산요원으로 구성된 'CBE 태스크 포스팀'을 결성하여 교육용 S/W 개발 작업에도 박차를 가하고 있다.

-태진전기

초박형 정현파 UPS개발

AVR / UPS 전문생산업체인 (주) 태진전기(대표 : 이호철)는 최근 UPS(무정전 전원공급장치)의 초박형 정현파제품을 개발, 시판에 들어갔다.

동사는 그동안 생산하고 있던 UPS 에이스 하이 시리즈가 다기능과 정현파 등의 장점을 갖고 있었으나, 가격이 비싼 단점이 있음에 착안, 2억여원의 연구개발비를 투입하여 개발에 성공한 것이다.

종래 국내의 초박형제품은 구형파방식을 채택, 사용함으로서 순간전원을 절체시 과도 피크전압과 고조파를 함유한 서지 및 노이즈전류가 발생하여 SMPS를 사용하는 컴퓨터 및 주변기기에 악영향을 주어 컴퓨터의 파손과 데이터의 손실을 가져왔다.

-럭키금성그룹

산전부문 4사, FA분야 집중 개발키로

금성산전, 금성계전, 금성기전, 금성하니웰 등 럭키금성그룹의 산전부문 4사는 89년도 공장자동화 및 산업전자분야의 연구에 총160억원을 투자하여 시스템, 전력전자, 메카트로닉스, 제어기기 등을 집중 개발키로 했다.

이와 같이 89년도 연구개발 투자계획은 지난해보다 30%이상이 늘어난 규모인데, 시스템분야에서는 약40억원을 투자하여 발전소용 광응용 계측제어시스템 개발, 열차제어 및 도로교통관제 등 데이터 전송시스템과 디지탈 분산제어시스템의 고기능화를 추진할 계획이다.

또한 전력전자분야에서는 약30억원을 투입, 수배전반용 디지탈 보호계전기와 전력계통 집중제어 표시장치 개발 및 전자식 배전반의 모델다양화에 주력키로 했으며, 메카트로닉스분야는 원통좌표 및 고속 수평다관절 로보트의 개발과 서보드라이브 유니트 등의 개발에 약50억원을 투자할 예정.

이외에도 제어기기분야의 고기능 PLC 개발 및 원격검침시스템의 신뢰도향상과 계측기기분야의 산업용 검사장비시스템 개발에 각각 20억원을 투자한다.

이와 같은 연구개발을 차질없이 추진하기 위해 산전부문은 통합연구소의 연구인력도 현재의 190명에서 250명 수준으로 늘리기로 했다.

-삼성그룹

최대의 인사개편 단행

삼성그룹의 정기인사개편이 1월 초순에 있었다.

이번 인사의 두드러진 특징은 삼성그룹의 인사사상 최대규모란 점과 고 이병철회장의 인사방침인 신상필벌주의에 벗어난 승진일도의 모습을 보이고 있다. 192명의 최대 승진자를 배출한 이번 인사가 갖는 또 하나의 중요한 의미는 신임 이회장이 추진해 온 책임 경영체제를 더욱 가속화했다는 점이다. 전자, 중공업, 엔지니어링 등 역점사업부문의 부회장제도 도입은 자율경영제도를 의미하며, 대규모 사업부문에 대한 대표이사제 신설이 책임경영의 실현을 대변해 주고 있다.

또한, 삼성전자의 승진규모가 49명으로 두드러지게 크고, 192명의 승진자중 113명이 엔지니어 계통이라는 점이 기술중시의 정책변화를 뒷받침해주고 있다.

-삼성데이타시스템

잎담배 수매 전산시스템 출원

삼성데이타시스템(대표 : 임경춘)은 지난 1월11일 입담배의 수량을 자동측정하여 계산하는 '입담배 수매 전산시스템'을 특허청에 실용신안 등록을 출원했다고 밝혔다.

동 시스템은 입담배의 중량 측정과 감정, 창고의 입고와 출고 현황을 자동적으로 처리할 수 있도록 하였다.

또한 동 시스템은 중앙제어시스템을 비롯하여 감정합의기, 전자저울, 계산서 인쇄기, 전원 시스템으로 구성되어 있다.

-삼성물산

CATV사업 본격 진출

삼성물산은 최근 호텔CATV 사업에 본격 참여, 스위스그랜드호텔, 서울프라자호텔 등에 CATV 시스템을 설치하여 방영에 들어갔다.

삼성물산이 이번에 설치한 시스템은 삼성전자 관련사가 개발한 첨단다기능 쌍방향 CATV시스템으로 재송신방송, 안내방송, 비상방송, 유로영화방송 등의 기능을 갖고 있으며. 향후 종합유선방송과의 연결 및 채널확장을 위해 최고 64채널까지 채널확장이 가능토록 했다.

삼성은 호텔 CATV사업을 계기로 국내호텔을 대상으로 CATV보급을 확대하는 한편 미국, 유럽 등에 CATV 시스템 수출을 본격화할 계획이다.

-STM

FM사업 비그룹사에도 확대

STM(대표 : 김영태)이 전산실 수탁관리사업(FM : Facilities Manageme-nt)을 비그룹사에도 확대하여 본격적으로 추진하기 시작했다.

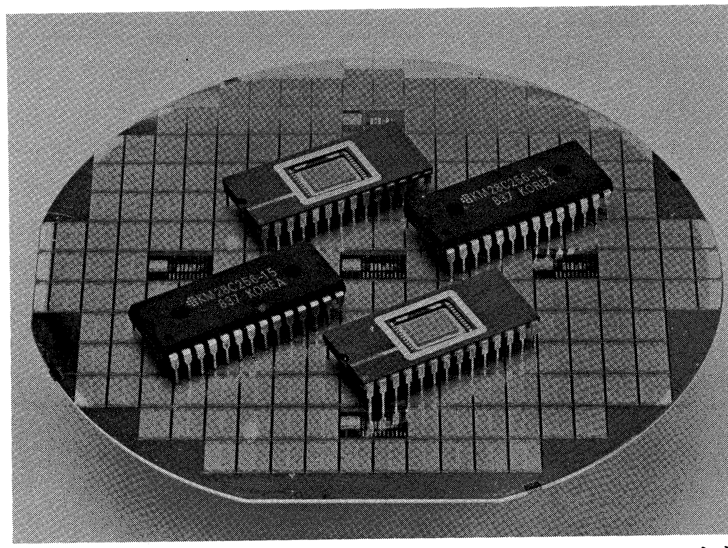
동사는 1차적으로 럭키금성계열의 26개와 전산실 수탁관리 장기계약을 체결하여 추진한데 이어 비그룹사로 전산처리업무 대행영역을 확대해 가고 있다.

STM은 지난해 동방유량, 미쉐린타이어 등과 FM에 대한 장기계약을 맺은 바 있으며 올해에는 영역확대 방침에 따라 증권, 보험 등의 금융기관과 전자, 전기 및 화학분야의 제조업체들을 상대로 FM사업을 본격 추진할 계획이다.

또한 해외정보처리사업에도 진출하

여 정보처리기술과 고급전산인력의 수출에 주력할 방침이다.

STM사는 럭키금성그룹이 미국의 FM전문업체인 EDS사와의 합작으로 지난 87년에 설립된 업체로서 초대형 컴퓨터 4대와 대형컴퓨터 50여대, 그리고 9백50명의 인원을 갖춰 정보처리 기술용역을 비롯 전산자원대여, 전산 시스템통합 및 전산실 수탁 관리업무 등의 사업을 해왔다.



-삼성전자

256K EEP롬 개발 성공

삼성전자(대표 : 강진구)는 지난 12월 순수한 자체기술로 256K EEP롬 (Electrically Erasable Programmable ROM) 개발에 성공하여 시제품 양산에 들어갔다.

지난 84년 16K EEP롬과 86년 64K EEP롬을 개발 생산한 동사는 256K EEP롬을 개발하기 위해 기흥연구소의 박형규박사외 6명이 87년 6월 개발에 착수하여 14개월만에 성공하게 된 것이다.

이번에 개발된 256K EEP롬은 단전이 되어도 정보가 남아 있는 롬의 특성과 단일 5V전원으로 리드/라이트 가능한 램의 특성을 동시에 갖춘 메모리로서 프로그램을 보드상에서 자유로이 바꿀 수 있어 고객이 사용하기에 편리하게 되어 있다.

또한 이 256K EEP롬은 64K EEP롬에 사용되지 않았던 에러검출 및 정정장치(ECC : Error Checking And Correction) 를 내장함으로써 에러가 발생된 정보를 자동으로 수정할 수 있으며 프로그램 수정횟수는 기존의 1만회에 비해 최대 10배 정도 늘릴 수 있는 장점을 갖고 있다. 아울러 프로그램 에러를 방지할 수 있는 기능도 갖고 있다.

특히 이 롬은 1.2㎛의 미세가공 기술이 요구되는 CMOS제품으로 신문지 2페이지에 해당하는 3만2천자의 기억용량을 갖추고 있으며 IC카드나 모뎀, PABX, 디지탈 TV, 프린터기기, 로보트 등 민생 및 통신산업용과 미사일, 항공기 블랙박스 등 우주항공기에 이르기까지 광범위하게 응용되고 있다.

한편 EEP롬의 세계시장 규모는 91년에는 3억5천만불 규모로 연평균 25% 이상 고성장할 것으로 예측되고 있다. 특히 동사는 이 EEP롬의 생산으로 향후 5년간 1억5천만불 이상 수출할 계획이다.

-삼성전자

모니터 등 정보기기 CCP승인 취득해

삼성전자(대표 : 강진구)는 최근 캐나다의 안전규격관장기관인 CSA(Canadian Standard Association)로부터 모니터와 터미널 등의 정보기기에 대해 CCP(Category Certification Program) 승인을 취득함으로써 향후 각종 컴퓨터 관련기기의 미주시장 석권에 중요한 전기를 맞게 되었다.

따라서 동사는 수출제품마다 테스트용 샘플과 제반서류를 CSA에 송부하던 절차를 생략하고 자체적으로 테스트를 시행할 수 있게 되었으며, KIMM (한국기계연구소)으로부터 받아온 사후관리까지도 직접 관장하게 되었다.

이로써 동사는 3개월 이상 걸리던 수출제품의 검사기간을 1주일로 대폭 단축할 수 있게 되었을 뿐 아니라 관련경비를 절감할 수 있게 되었다.

캐나다의 CCP제도는 단순한 제품의 품질평가 뿐만 아니라 테스트설비 및 관련 엔지니어의 자질과 완벽한 생산공정, 계측기기의 관리상태와 소비자 보호장치 등이 CSA가 요구하는 적정수준에 달해야 하는데 그 기준이 무척 까다로와 취득이 어려운 것으로 알려져 있다. 이를 동사에서 국내는 물론 아시아에서는 최초로 승인을 받게 된 것이다.

-삼성전자

NCR사와 기술교환계약 체결

삼성전자(대표 : 강진구)는 12월26일 ASIC(Application Specific IC : 특정 용도 IC)분야에서 세계적인 기술수준을 보유하고 있는 미국의 NCR사와 기술교환(Cross License)계약을 체결.

이는 그동안 국내 반도체업계의 기술계약 형태가 일방적인 기술도입 형태를 취한데 비해 그러한 관계가 아니라 상호 기술교환 협정이라는 점에서 의의가 크다.

이번 계약으로 삼성전자는 향후 6년



간 NCR 스탠다드 셀과 설계자동화 소프트웨어 기술을 도입할 수 있게 되었다. 따라서 동사는 첨단 메모리 반도체 중심으로 추진해 왔던 사업을 세계 수준의 ASIC기술을 도입하게 됨으로써 초기 단계의 ASIC 사업에서 탈피하여 고부가가치 ASIC제품을 공급할 수 있게 되어 제품경쟁력을 확보하게 되었을 뿐 아니라 최근 성장속도가 빠른 주문형 반도체사업에도 본격적으로 진출할 수 있게 되었다.

아울러 NCR사도 삼성의 S램 기술을 이전 받음으로써 이를 기반으로 하여 메모리사업을 적극화할 수 있게 되었다. 이 NCR사는 1888년 설립되어 현재 반도체, 컴퓨터제품 등에 주력하고 있다. 이 특히 ASIC분야중 스탠다드 셀에 있으며 세계 수준의 기술을 보유하고 있다.

-삼성종합기술원

DTP용 워크스테이션 자체 개발

삼성종합기술원은 CAD / CAM이나 전자출판시스템에 적합한 32비트 엔지니어링 워크스테이션을 자체기술로 개발하는데 성공했다.

동 워크스테이션은 인텔의 80386 마이크로프로세서를 채택하고 OS로서 유닉스를 지원하고 있어 멀티유저와 화면분할 등의 기능을 갖고 있다.

엔지니어링 워크스테이션은 그간 미국 선마이크로사, 아폴로컴퓨터, DEC사 등에서 미니급이나 수퍼마이크로급 컴퓨터수준의 고가품을 공급해 왔으며, 32비트 퍼스컴수준의 저가 386기종은 작년 10월 선마이크로사가 처음으로 상품화하였다.

삼성전자에서 상품화하게 될 동 시스템은 자체개발한 그래픽보드를 사용함으로써 가격인하가 가능하게 되었다.

-쌍용컴퓨터

BPCS 관련 기술도입 승인받아

쌍용컴퓨터(대표 : 최병환)는 최근 과학기술처로 부터 종합생산관리 소프트웨어인 BPCS(Business Planning and Control System) 관련 기술도입 승인을 받았다.

1988년7월1일 정부의 기술도입 승인요건의 강화이후 소프트웨어 관련기술이 도입된 것은 쌍용컴퓨터의 이번 BPCS 관련 기술도입이 처음이다.

과학기술처는 종합생산관리 소프트웨어 개발이 현재의 국내기술로는 어려운 상황일 뿐 아니라 이와 관련한 기술이 전수되었을 때 한국형 종합 생산관리 소프트웨어의 개발이 가능케 된다는 효과가 있으며 기술제공자인 미국 SSA(System Software Associates) 사의 기술력과 BPCS의 기능이 탁월하다는 점 때문에 기술도입을 승인한 것으로 보인다.

이에 따라 쌍용컴퓨터는 기술제공사인 SSA사로부터 BPCS 관련 소스 및 기술자료를 제공받아 이를 우리 실정에 맞는 시스템으로 공급할 예정이다.

쌍용컴퓨터가 도입한 BPCS는 제조 및 유통분야 기업의 경영관리를 위한 통합소프트웨어로서 생산, 판매, 재무 분야와 관련된 20여개의 모듈이 상호 유기적으로 구성되어 있으며 필요에 따라 모듈별로 단계적 도입이 가능한 것이 특징이다.

-쌍용컴퓨터

알랜 브래들리와 SI계약 체결

쌍용컴퓨터(대표 : 최병환)는 최근 공장자동화(FA)기기 전문업체인 알랜 브래들리(Allen Bladley)와 국내시장에 있어서의 FA프로젝트에 컨소시움을 형성하여 공동개발 및 공급을 하기 위한 관계수립의 일환으로 QIM (Quality Information Management)기기에 대한 시스템 인티그레이터 (SI : System Integrator)계약을 체결하였다.

이 계약의 주요 내용은 동사가 알랜 브래들리사의 영상처리기기인 비전시스템을 비롯하여 바코드, RFID(Radio Frequency Identification)와 자체개발 FA용 소프트웨어를 통합하여 품질정보관리(QIM)시스템을 구축한다는 것이다.

따라서 동사는 알랜 브래들리의 제품 및 FA관련 노하우에 자사의 소프트웨어 노하우 등을 접속한 고부가가치의 FA솔루션을 공급하게 되었을 뿐 아니라 SI로서 국내기업의 FA추진에 적합한 토탈시스템 공급체계를 갖추게 됐다.

반면 알랜 브래들리사는 동사를 통해 극내에 처음으로 자사제품의 SI를 이룰 수 있게 됐다.

한편 동사는 이번 89년4월부터 종합 CIM(컴퓨터 통합생산)체계를 도입하여 이를 사업화할 피라미드 인티그레이터(Phyramid Integrator)로서 FA 사업영역을 더욱 확대할 계획이다.

–스카이시스템
GP-IB 보드 국산화 성공

스카이 시스템(대표 : 김현진)은 컴퓨터 통신방식의 일종인 GP–IB(범용 인터페이스 버스)를 자체로 개발하여 외국제품의 50% 가격으로 출하하였다.

동 시스템의 특징은 NEC나 HP 기종은 물론 IBM 호환 퍼스컴에도 장착할 수 있는 것이며 이 제품은 주로 퍼스컴의 슬롯 가운데 한곳에 연결하여 컴퓨터의 외부 제어장비를 연결해서 장비를 원격으로 조종하고, 계측데이터를 분석처리하는 데 이용된다.

동 시스템은 HP, NEC, 대만 제품이 거의 장악하고 있는 국내 실정을 비추어 볼때 최초의 국산화 제품으로 업계의 관심의 대상이 되고 있다.

–신한전산
통신회선을 이용한 S/W 개발 개시

(주)신한전산(대표 : 유영준)은 일본의 정보처리 서비스 회사인 (주)일본 비지니스 컨설턴트와 국제 통신회선을 이용하여 소프트웨어 개발업무를 개시하였다.

이는 신한전산에 설치한 소프트웨어 개발용 TSS단말과 일본 비지니스 컨설턴트의 중정분공장의 대형컴퓨터(히다찌 M–660K)를 국제통신회선(전용회선)으로 연결하여 신한전산이 일본 비지니스 컨설턴트로 부터 수주한 소프트웨어 개발업무를 이 TSS단말을 이용해서 개발하는 것으로 되어 있다.

이에 따라 양사는 동일한 환경으로 소프트웨어를 개발할 수 있게 되어 작업 효율의 향상, 비용절감으로 큰 성과를 올릴 것으로 전망하고 있다.

–쟈네트시스템
인텔리전트 모뎀 개발

모뎀 전문업체인 쟈네트시스템(대표 : 고시연)은 인텔리전트 모뎀 4개 기종을 자체 개발하여 시판에 나섰다.

동 시스템은 모두 전용회선 모뎀으로 CCITT규격 V.29, V.27bis와 체신부의 표준규격을 채택한 제품이다.

이들 인텔리전트 모뎀은 동사의 기존모뎀에 비해 크기를 절반 정도로 줄인 콤팩트형 제품으로 자체진단기능·원격모뎀제어·원격모뎀상태표시기능을 갖추고 있다.

한편 모뎀상태 및 기능표시를 위해 액정디스플레이(LCD)를 채용함으로서 메뉴방식으로 기능을 선택할 수 있어 사용자의 편의를 돕고 있다.

동사에서는 자체 개발한 인텔리전트 모뎀의 복제방지를 위해 이미 과기처에 관련 소프트웨어를 등록해 두고 있다.

–한국컴퓨터통신
DOS용 네트워크 S/W 개발

한국컴퓨터통신(대표 : 최은섭)은 대형컴퓨터간의 통신에 주로 적용되고 있는 TCP / IP 통신 프로토콜방식을 응용한 DOS용 PC / TCP 네트워크 소프트웨어를 개발하였다.

이에 따라 서로 다른 메인프레임을 비롯, 각종 미니컴퓨터·워크스테이션·퍼스컴들간의 정보교환이 가능하게 되었다.

특히 동 시스템은 한글을 지원할 뿐만 아니라 어떤 종류의 컴퓨터이건 TCP / IP가 탑재되어 있으면 별도의 통신장비를 부착하지 않고도 3270에뮬레이터 등의 간단한 접속방법에 의해 한글 및 영문데이터를 동시 접속할 수 있다.

또한 이더네트·IBM 토큰링·프로네트·×25·RS232C 등 각종 네트워크의 지원이 가능하여 사용자가 통신망을 구성 확장할 수 있으며, 보안장치를 내장하여 각종 정보의 유출을 사전에 방지할 수 있다.

–한국팔란티어
PWP 패키지 수출 개시

국내 최초로 한글 패키지 S / W가 정식 수출되기 시작하였다. OA용의 패키지 S / W 개발 전문업체인 한국팔란티어(대표 : 김춘)에서는 업무용 PC에서 사용하는 한글, 한자 워드프로세서 프로그램인 PWP를 미주지역에 수출키로 하고 지난 14일 테스트 마케트를 위해 첫 선적에 들어갔다. 수출선은 뉴욕 소재의 재미 한국인이 경영하는 Tricom Inc.라는 컴퓨터 업체인데 동사에서는 미국 전역의 한인 교포 및 한국어를 전공하는 미국인을 상대로 연간 1,000개 정도의 판매목표를 정하고 있다고 한다. 팔란티어측은 이번 계약을 시작으로 하여 올해에는 캐나다, 유럽, 중국등지의 한인 사회에도 PWP를 공급할 계획을 세우고 다각도로 이를 추진중에 있다.

–엘렉스컴퓨터
권순덕씨 전문이사로 영입



(주)엘렉스컴퓨터(대표:이윤기)는 삼보컴퓨터 권순덕 상무를 1989년 1월4일부로 전무(사진)로 영입하였다. 이는 엘렉스컴퓨터의 주목표인 DTP시장을 강화하기 위한 일환이라 한다.

유진양행, 후발 3개 업체와 함께 4파전 벌일 각오

IBM호환기 메이커이자 EISA의 "The Gang of Nine"의 3대 핵심멤버중 선발대라고 할 수 있는 미 AST 리서치사가 최근 기존의 디스트리뷰터인 유진양행을 뒷전으로 하고 국내의 소프트웨어전문업체인 ICS와 옴니테크, 다우기술 등 3사의 공동 디스트리뷰터 계약을 체결함으로 인해 또 다시 좁은 국내시장에서의 출혈경쟁을 재현시키고 있다.

이러한 미 AST 리서치사의 이중계약과정은 미국의 본사에 의한 것이 아니라 홍콩에 있는 아시아지부에 의한 단독결정인 것으로 알려지고 있다.

지난 12월 17일 AST 리서치사의 극동담당 총책임자인 필립 황씨가 내한하면서 이루어진 이번 계약은 내한 당시 기존 국내 대리점계약을 맺고 있던 유진양행에는 한마디의 사전 양해도 없이 이루어졌다.

이 계약으로 ICS는 주공급체로, 옴니테크는 한글내장카드를, 다우기술은 5250 터미널의 한글 지원 등을 각각 맡아 하게 되었다.

이러한 이중계약이 이루어진 동기는 84년부터 유진양행을 통해 국내시장에 AST 리서치사의 제품을 공급하였으나 유진양행의 기술수준이 AST 리서치사가 생각하기에 아직까지는 부족다고 판단해 단지 스캐너와 레이저 프린터 등 일부품목만 국내시장에 공급하였다. 그러나 지난 88년 9월 AST리서치사가 프리미엄 386SX PC를 발표하면서 국내시장에 이를 공급하여 현재 국내시장에 깊이 잠식해 있는 IBM PC에 도전장을 내놓을 계획이었고 독단적으로 유진양행이 이를 소화하지 못할 것으로 판단하고, 기존의 유진양행에는 사전 통보없이 국내에서 이를 소화할 만한 업체를 물색하게 된 것이 지금의 이중계약을 맺게 된 동기가 되었다.

이러한 이중계약을 뒤늦게 안 유진양행측의 한 관계자는 "84년부터 87년말까지 서류상으로 독점계약을 맺었으나 그뒤부터는 서류상 계약을 맺지 않았어도 오랜 거래로 미국 본사의 AST 리서치사와 어느 정도는 신용이 두터워졌다고 판단됐고, AST 리서치사의 제품공급은 당연히 유진이 한다고 생각하고 있는 상황에서 이런 일은 전혀 예상치 않은 것이다"라고 하고 있다.

또한 최근 AST 리서치사가 32비트 386X를 개발한 사실을 안후 언제까지나 주변기기만을 공급할 수 없는 입장이고 해서 유진양행측은 나름대로 한글카드내장을 위한 연구를 진행중이었다.

소자본으로 주로 AST리서치사의 제품을 공급해 온 유진으로선 이번 일로 AST사와의 관계가 막을 내린다면 받는 타격은 크다.

앞으로 유진양행은 ICS측과 한번쯤 대면을 생각하고 있으나 아직까지는 시기상조인듯 잠시 앞으로 전개될 일에 대해 예의 주시하고 있다.

반면 ICS사의 한 관계자는 "계약 당시 국내에 미AST 리서치의 제품을 유진양행에서 공급하고 있었다는 사실을 몰랐으며, 이미 계약을 끝냈고, 제품공급을 위한 준비단계인 상황에서 알았다 해도 어쩔 수 없는 상황"이라고 주장.

그러나 ICS사는 AST리서치사의 전품목공급을 계약했기 때문에 앞으로 국내에서 ICS사외에 AST리서치사의 품목을 공급하고자 하는 업체는 반드시 ICS를 거쳐야 한다고 강하게 어필하면서 유진양행도 이에 적용되며, 국내에 딜러를 둘지의 여부는 검토중이나 만일 ICS사측에서 이를 확실하게 결정한 뒤 유진양행측에서 이에 응한다면 좋은 방향으로 이끌 용의가 있다는 입장을 내세우고 있다.

여하튼 이런 이중계약이 자의건 타의건간에 국내업체들끼리 말려들어 치열한 접전을 벌인다면 그것은 결국 우리내 모두의 손해이다. 그렇다고 경쟁이 없는 사회로 몰아가자는 것은 아니다. 다만 경쟁을 하되, 좀더 신사적이고 양심적인 화인 플레이를 하자는 것이다.

그러나 현시점에서는 어느쪽이 옳고 그르다고 손가락질하며 탁상공론을 벌이기 보다는 이미 벌어진 일을 전제로 가장 합당한 방법을 찾아 양측 모두 큰 피해없이 해결되길 바란다.

〈글. 김현정기자〉

국외소식

미국 컴퓨터업계
향후 2~3년이내 바이러스산업 호황 이룰듯

미국은 전통적으로 「타인이 절대로 모방할 수 없는 예능이나 기술」에 대해서는 관용을 넘어서 적극적으로 찬사를 아끼지 않는 나라이다.

예를 들어 선악의 윤리를 다소 벗어난 일을 하고, 비지니스상의 부적당함이 발생하여도 그일에 대한 충격을 감동으로 미화시켜 현혹하는 경우가 있는 것 같다.

지난해 말 23세의 코넬대학원생 로버트 T. 모리스 Jr.가 미국방부의 컴퓨터 네트워크인 ARPANET에 바이러스를 잠입시켜 전국의 여러 연구기관, 대학에서 가동중인 6천여대의 컴퓨터를 혼란에 빠트린 사건이 발생했다.

법적으로 하면 모리스는 중대한 컴퓨터범죄자로서 벌을 받아야 할 것임에도 불구하고 미국의 일반적 반응은 「죄와 벌은 다르다」고 하는 표현방법으로 나타나고 있다. 다시 말하면 컴퓨터의 귀재 모리스가 창조한 바이러스가 거대한 컴퓨터 네트워크 시스템의 맹점을 찌른 도전은 오히려 쾌거라고 해야 한다는 아이러니컬한 평가가 대세를 차지하고 있는 것이다.

모리스의 바이러스사건을 컴퓨터범죄로 취급하지 않고 오히려 핵커기질의 양성이라는 측면에서 적극 평가하고자 하는 흐름이 왜 생겨났으며, 존재하는 것인가에 세인의 관심이 집중되고 있다.

모리스사건은 어떤 의미에서는 일어날 것이 당연히 예상된 사건이다. 모리스가 자라온 가정환경과 부친과의 관계가 그것을 말해주고 있다. 부친 로버트 T. 모리스 Sr.(56세)은 벨연구소에서 26년간 Unix 개발에 참여한 과학도였으며, 현재에는 일반적으로 그 존재가 베일에 가려진 정부첩보기관(National Security Agency)의 상급간부이며, 암호제작 독해의 전문가이다. 모친도 또 MIT출신의 재원이였다.

이렇게 우수한 두뇌와 왕성한 독립성의 혈통을 이어받은 모리스가 부모의 자문을 얻어 습득한 「컴퓨터 시큐리티기술」을 사용하여 거대한 국방망에 도전한 결과가 이번 사건의 전모이다.

이번 사건은 많은 미국인에 쾌감과 동경을 줄 수 있는 것이다. 그러나 컴퓨터 바이러스방지에 부심하고 있는 피해자측에서 보면 그 피해는 심각하다. 엄격한 의미에서는 이러한 바이러스 범죄는 벌을 받아야 하며 그와 동시에 유효한 바이러스방지책도 적극적으로 검토되어야 한다는 것이 이번 사건의 큰 선물이다(표 1 참조).

미국에서는 아직 컴퓨터 바이러스범죄를 처벌대상으로 하는 법률이 존재하지 않는다. 적용관련도가 높은 현행법은 대컴퓨터 비합법 액세스와 방해를 대상으로 한 「컴퓨터범죄 악용방지법」(1986년 시행) 밖에 없다.

그러나 이 법률하에서는 최고 1년의 징역(또는 최고 25만달러의 벌금—죄가 명확한 경우는 최고20년의 징역)이라고 하는 벌칙규정 밖에 없다.

컴퓨터 바이러스가 많이 발생하여 완전히 사회문제화된 것은 지난해부터였다. 그런 의미에서 지난해는 「컴퓨터 바이러스대책의 원년」이라고 할 수 있다.

최근 미국에서는 컴퓨터 바이러스예방을 위한 방지대책의 수요와 공급이 급격히 증가하고 있다.

〔표 1〕 컴퓨터 바이러스 예방대책예

- 소프트웨어는 신용있는 점포(회사)에서 패키지제품을 구입한다.
- 패키지를 열람한 후 백업 카피를 만들어 안전한 장소에 보관한다.
- 사용전에 소프트웨어오염의 유무를 필히 체크한다. 방법은 ① 시스템에서 떨어진 컴퓨터에서 테스트하며, ② BBS나 PDS의 소프트웨어에 대해서는 특수한 검역처리를 한다.
- 1달에 1번은 시스템 소프트웨어의 백업 카피를 만들어 최저 1년간 보존한다. 이것이 시한폭발하는 바이러스에 대한 유효 예방책이 된다.
- 프로그램의 규모(용량)변화 체크를 정기적으로 실시한다. 바이러스 오염상태를 탐지하는 수단이 된다.
- 바이러스 오염증상이 있는 소프트웨어는 발견 즉시 제거하며, 관련 데이터는 카피한다. 감염문제가 있을 때는 외부통신 네트워크와의 접촉을 끊는다.
- 컴퓨터 시스템과 네트워크의 이용은 관계자 이외에는 출입을 통제한다.

〔표 2〕 컴퓨터 바이러스 대책제품/서비스 주요회사

회 사 명	소 재 지	제품/서비스
Digital Pathways Inc.	Mountain View, CA	데이터 시큐리티 서비스
Interpath Corp.	Santa Clara, CA	컴퓨터 바이러스대책 상담
LeeMah DataCom Security Corp.	Hayward, CA	전화호출 확인시스템
Remote Technologies Inc.	St. Louis, MO	바이러스 대항제품 바이러스 오염세정 서비스
United Software Security Co.	Vienna, VA	데이터 시큐리티 서비스
International Information Integrity Institute(SRI 부속기관)	Menlo Park, CA	컴퓨터 불능(오염)문제 조사

이러한 종류의 재품 서비스를 제공하는 전문회사가 지난 1년간 35개 업체로 급증하였다. 또한 향후 2~3년내로 컴퓨터 바이러스산업이 소프트웨어 분야의 주요시장으로 등장할 것이라고 예측되고 있다(표 2 참조).

그러나 바이러스감염원이 현재 전자게시판 시스템(BBS)이나 PDS(Public Dowain Software) 및 쉐어웨어 등에 집중하고 있어 BBS의 사용을 대기(정지)시킨다든지, 회사와 자택에서 같은 플로피 디스크의 사용을 금지시킨다든지 하는 움직임이 나오고 있다.

또한 소프트웨어 메이커간의 치열한 경쟁속에서 경합상대선의 소프트웨어에 바이러스를 친입시키는 움직임(가능성)을 예측하는 시장분석가도 등장하고 있다. 컴퓨터 바이러스의 문제해명과 시장형성은 앞으로 많은 논란을 불러일으키며 진전될 것이다.

AT&T
HP컴퓨터의 OEM계약 확장

휴렛팩커드사와 AT&T는 최근 AT&T 의 DSP32C 디지탈 신호처리기(DSP)를 위한 포괄적인 개발 툴을 공동으로 지원 개발한다는 협정과 더불어 88년초에 서명하였던 HP컴퓨터의 주변기기에 대한 OEM계약을 확장한 것으로 알려졌다.

DSP에 대한 협정에 따르면 앞으로 HP측은 AT&T의 DSP32C 마이크로칩에 상응하는 리얼타입 DSP에뮬레이터 신형을 설계하여 시장에 판매하게 될 것이고, AT&T측에서는 PC나 데스크탑 컴퓨터와 HP의 신형 에뮬레이터가 완전 호환이 가능하도록 통합 소프트웨어 개발 툴을 제공할 것으로 알려지고 있다. 따라서 AT&T는 소프트웨어 개발을 위하여 C컴파일러, 어셈블러 및 링커를 최적화 할 계획이다.

HP의 신형 DSP에뮬레이터는 HP647000 에뮬레이터시리즈 중의 하나이다. 이 시리즈는 단일 칩 논리분석기(Logic Analyzer) 기술을 이용하여 고도의 집적화를 가능하게 한 것이다. 따라서 사용자측에서 본다면 컴퓨터에서 MS-DOS나 HP-UX 운영체제 어느 것으로도 운용이 가능한 에뮬레이션을 선택할 수 있게 된 것이다.

AT&T의 DSP32C 마이크로칩은 32비트 부동소수점 디지탈 신호처리기로서 50MHz에서 운행되며 1초에 2천5백만번의 부동소수점 연산을 수행할 수 있다. 그리고 고도의 0.9미크론(Micron) 더블 CMOS로 제작되어 있다.

또한 이 칩은 그래픽이나 영상처리, CAD/CAM, 공학적인 계산 및 고종단(High-end) 텔리커뮤니케이션 등 그 응용범위는 매우 넓다.

AT&T는 또한 지난 87년 5월 HP의 5.25인치 디스크 드라이브(모델 9753S) 에 대한 협정을 한데 이어 다시 이에 대한 OEM계약 확장 협정을 하였다.

그 협정 내용은 AT&T가 HP의 5.25인치 윈체스터 디스크 드라이브(모델97536D/T)와 0.5인치 테이프 드라이브(모델88780A)를 동사의 마이크로 컴퓨터 생산라인에 부가시킬 것이라는 것이다.

이 협정에 대해 AT&T의 3B2 생산품 경영의 지역 매니저인 Ben N. Butler씨는 "우리는 품질과 신뢰도에 큰 비중을 두고 있기 때문에 OEM공급자도 우리와 동일한 생각을 갖고 있는 업체이어야 합니다. HP가 바로 우리의 이러한 바램을 충족시킬 만큼 우수한 품질과 지원해 주기 때문"이라고 말했다.

HP 97536S/D/T 5.25인치 디스크 드라이브의 용량은 포맷한 상태로 323M바이트이며 SCSI(Small Computer System Interface)를 내장하고 있다. 0.5인치 테이프 드라이브에는 6250bpi (Bit-Per-Inch) 그룹코드식 레코딩과 1600bpi 퍼텍(Pertec)호환 인터페이스를 내장하고 있다.

미, 터미널시장
ASCII 터미널에서 X터미널로 세대 교체

일반적으로 터미널이라 하면 네트워크 환경의 메인프레임이나 미니컴퓨터를 여러 유저가 공유하기 위한 디바이스이다.

터미널을 제조하고 있는 메이커는 대부분이 일본이나 NIES국가들로서 싸게는 1백달러 정도로 구입할 수 있다. 종래에는 DEC사양의 ASCII터미널이 사실상의 업계표준규격으로 되어 있었으며, 그래픽스 어플리케이션에 대응하는 것은 수가 한정되어 있었다.

그러나 최근 터미널의 중심이 되는

미니컴퓨터나 워크스테이션의 UNIX 환경하에서 미국 매서츄세츠공과대학이 개발한 X윈도우가 급속히 보급되고 있으며, 이에 따라 터미널이라고 불리우는 그래픽스를 근간으로 하는 터미널의 개발이 활기를 띠고 있다.

Visual Technology사 (미국 매서츄세츠주)는 「640X」라 불리우는 X터미널을 개발하였다. 동 터미널은 1024×800 도트의 14인치 모노크롬 디스플레이, 키보드, 마우스로 구성되며, 가격은 1995 달러. CPU에 68000을 채용하고 X윈도우 처리를 단말측에서 제어할 수 있다.

그리고 UNIX+X윈도우를 탑재한 퍼스널컴퓨터나 워크스테이션에 직접 혹은 네트워크를 통해 접속할 수 있다.

또 Acer Counterpoint사 (미국 캘리포니아주)는 「Xebra 모델100」이라고 불리우는 X터미널을 개발중이다.

이 두모델 모두 모노크롬표시지만 양사는 올해중에 컬러표시판 X터미널을 발표할 예정이다.

현단계에서는 X윈도우 대응의 어플리케이션은 아직도 수가 적다. 그러나 OSF(Open Software Foundation) 등이 X윈도우를 사실상의 표준 윈도우 시스템으로 채용하고 있으며, 올해에도 X윈도우 대응소프트웨어가 관련업체에서 계속 발표될 것으로 전망된다.

X윈도우의 보급에 수반하여 ASCII 터미널의 대부분이 X터미널로 이행하는 것도 예측된다.

IBM사
후지쯔와의 저작권 분쟁 끝내

미국IBM과 후지쯔의 OS를 둘러싼 저작권분쟁에 관해 미국중재협회(AAA)는 IBM과 후지쯔에 대해서 최종결정을 내렸다.

이것에 의하면 후지쯔가 IBM에게 지불하는 일괄지불금(화해금)은 약 3억9천6백만 달러이며, 이제까지 지불된 금액을 제외한 지불액은 약 2억3천7백만 달러이다.

후지쯔와 IBM의 이 분쟁은 1985년에 IBM이 「후지쯔는 83년에 맺은 소프트웨어 저작권에 관한 양사의 협정을 위반, IBM의 권리를 침해했다」고 하여 AAA에 중재를 신청하여 표면화되었다.

이번 AAA의 재조정은 구속력을 가지며, 더구나 공소제도가 없기 때문에 IBM의 제소이래 3년만에 분쟁이 매듭지어진 것이다. 이번 재정의 요지는 다음과 같다 (표 3 참조).

〔표 3〕 주요 재정내용

82년 10월	IBM, 후지쯔에 프로그램 저작권 침해로 항의
83년 7월	IBM과 후지쯔, 83년 화해계약
83년~85년	IBM, 후지쯔에 계약위반을 항의
85년 7월	IBM, 후지쯔가 83년 계약을 위반하여 AAA에 제소
12월	AAA의 심리개시
86년 10월	IBM과 후지쯔, 약식동의
12월	지정프로그램 계약(86년 계약)
87년 2월	SF 제도를 정한 워싱톤 계약
3월	SF 세부항목에서 합의 불성립
9월	AAA, 「명령」, 위성통신 기자회견
88년 6월	SF 세부항목 결정
11월	AAA, 「최종 결정」

● 정보의 액세스 정도에 따라서 중재인이 정하는 액세스료는 '89년도에 대해서는 2천5백67만달러에서 5천1백34만달러 정도. 그러나 사실상의 화해금인 일괄지불금은 약 3억9천6백만달러이다.

● 후지쯔는 일괄지불금을 지불하는 것으로 지난 83년도에 합의된 각 분기마다의 지불을 면하게 되고, 일정한 조건에 따라 IBM의 프로그램을 해석하여 호환기개발을 할 수 있게 됐다. 반대로 IBM은 소프트웨어저작권이 보장될 뿐만 아니라 법률 등에 명확한 규정이 없는 부분에 대해서도 보장을 받을 수 있다.

● 후지쯔는 IBM의 소프트웨어를 해석하는 장소인 SF(Security Facility)를 일본에 설치한다. 장소나 시간은 아직 결정되어 있지 않다. SF의 시설관리는 중립적인 관리자를 선정, 운영할 것이다.

● SF내에서 후지쯔는 IBM의 프로그램에 대해서 AAA가 정한 범위내에서 액세스할 수 있다. 단, 대상 프로그램은 인터페이스정보에 한한다. 즉, 그 프로그램이 어떻게 실행되는가가 아니고 무엇을 하는 것인가 만을 알 수 있다.

● IBM기종과 후지쯔기종간의 네트워크나 데이터공용을 위한 정보는 1994년 1월1일까지 출하되는 IBM소프트웨어의 정보에서 얻어진다.

● AAA의 중재인 관리권은 2002년 11월 29일까지 양사에서 발생되는 모든 분쟁에 효력을 발생한다.

미국 Architech사
OS/2용 윈도우 툴 키트 개발해

Architech사 미국(뉴욕주)는 OS/2용의 윈도우 툴 키트(Window Tool Kit)를 개발했다. 기존의 윈도우 툴키트인 선 마이크로시스템사의 네트워크

대응 윈도우환경이지만, OS / 2용 NeWS는 Architech사가 Sun사에서 라이센스를 받아 OS / 2에 이식한 것이다.

Mac Ⅱ의 A / UX환경에서는 Grasshopper Group사(캘리포니아주)가 MacNeWS를 이미 개발하였지만, NeWS / 2는 OS / 2용으로 주목되고 있다.

이 툴키트를 사용하여 개발자는 포스트스크립트로 기술한 텍스트나 그래픽스를 사용하는 OS / 2 대응의 어플리케이션 개발에 효과를 얻을 수 있다. 또 텍스트베이스의 OS / 2 어플리케이션을 변경하지 않고도 NeWS의 윈도우내에서 실행하는 것이 가능하다.

동사는 NeWS V1.1의 이식을 거의 완료하고 있으며, 이식이 완료되면 네트워크내의 서버 / 크라이앤드간에서 그랙픽스의 분산처리를 한다는 NeWS 본래기능이 OS / 2상에서 실현된다는 말이 된다.

Architech사에 의하면 NeWS / 2는 OS / 2의 프리젠테이션 매니저와 메모리내에서 공존할 수 있어 경합관계를 피하고 있다고 하지만, NeWS / 2가 1.5M바이트의 메모리를 점유하기 때문에 현실적으로는 프리젠테이션 매니저와 배타관계로 될 가능성이 있다.

인텔사

GE/RCA의 DVI부분 인수

인텔사가 GE / RCA의 DVI부문을 매수했다. DVI란 지난 87년에 개최됐던 CD-ROM컨퍼런스에서 극적인 등장을 하여, '88년의 동컨퍼런스에서는 리얼타임 콤프레스기능이 추가되는 등 차세대 퍼스널컴퓨터용 디스크탑 비디오 / 오디오시스템이다.

이 DVI의 원개발원인 GE / RCA사는 지난 88년10월 DVI부문을 인텔사에 매각함으로서 앞으로 DVI부문을 Intel Princeton Operation이 맡게 된 것이다.

인텔사는 칩메이커로서 DVI와 같은 멀티미디어기술의 도입을 모색하여 왔으므로 이번의 매수는 당연한 과정이라고 할 수 있다.

Intel Princeton Operation에서는 DVI칩의 디자인을 전면적으로 재검토함과 동시에 소프트웨어 개발툴 등을 정비하여 금년도부터 판매를 개시한다고 한다. GE / RCA가 지금까지 개발한 DVI시스템은 IBM PC / AT용 뿐이지만, 인텔은 앞으로 68000계 CPU용도 포함한 칩세트나 보드류를 개발할 계획이다. 이중에는 Mac Ⅱ용 DVI보드로 포함하고 있다.

Lotus사는 이번 인텔사의 DVI부문 매수를 호의적으로 보고 있다고 전해지며, Mac커뮤니티에 가까운 개발자들도 이를 환영하고 있다. 인텔의 전면개입으로 DVI어플리케이션의 개발에 한층 박차가 가해질 전망이다.

일본 통산성

레이저 개발 계획, 본격 추진키로

최근 레이저기술이 급속히 발전함에 따라 여러가지 신형레이저가 속속 발표되고 있으며, 특히 엑시머레이저의 등장에 따라 자외영역까지의 레이저광이 그 이용분야를 확대해 가고 있다.

레이저는 광의 빔을 집속할 수 있기 때문에 부분적으로 초고온을 가할 수 있고, 종래의 여러가지 에너지와는 전혀 다른 새로운 에너지로서 생산프로세스기술 등을 혁신시킬 수 있는 가능성을 갖고 있다.

레이저의 이용가능 분야로는 재료가공, 재료표면개질, 화학합성, 화학분리, 세포처리, 의료기기, 반도체 등이며, 그 파급효과는 모든 산업에 걸쳐있다고 추정된다.

유럽에서는 유레카계획의 일환으로 서독의 후라운호퍼연구소 등을 중심으로 레이저기술 개발의 대형 연구프로젝트가 진행되고 있으며, 미국에서도 군사용과 산업용레이저에 대한 연구가 진행되고 있다.

일본의 경우에도 지금까지는 레이저기술 분야에서 구미에 비해 답보한 상태였으나 최근들어 레이저에 대한 각종 연구보고가 쏟아져 나오고 있다. 그러나 일부대학이나 업체를 중심으로 연구가 진행되기 때문에 활발하다고 할 수는 없다.

일본 통산성 공업기술원에서는 이와 같은 레이저기술 개발의 세계적 동향과 일본 국내에 있어서의 실태 등 레이저이용 기술연구의 가속을 위해 새로운 시책을 구상, 예산편성에 결부시키고 있다.

일본 통산성의 레이저 개발 계획은 엑시머레이저, CO레이저, 탄산개스(CO_2)레이저, 요소레이저 등 최첨단의 레이저장치를 한 장소에 집중하여 정비하고, 다수 연구자가 공동이용할 수 있는 시설, 즉「레이저 응용공학센터」를 설립하는 것이다.

이는 제도적으로는 '88년10월에 발족한 신에너지 산업기술 종합개발기구(NEDD)의 연구기반 정비사업(제3 센터방식에 의한 시설정비)에서 행하는 것이며, 현재 금년도 예산요구의 알맹이중 하나이다.

통산성은 이같은 구상의 실현을 위해 지난 88년9월 (재)공업개발연구소에 레이저 응용공학센터 검토회를 설치, 레이저기술의 향후 동향을 전망하고, 센터에 어떤 설비를 갖추어야 하는가에 대한 구체적 검토를 시작하였다. 동년 12월에는 검토회의 중간보고를 정리, 예산안을 편성하는 금년 3월까지 최종보고를 할 계획이다.

동 시설물은 89~91년까지 3개년에 걸쳐 건설될 계획이며, 초기투자액은 30억엔 정도이다. PC

서클동정

●다솜
컴퓨터를 통하여 사랑을 나누는 아이들

국민학교, 중학교, 고등학교 학생으로서 컴퓨터 통신을 이용하여 자신들의 이야기를 나누는 다솜. 지난 88년 10월에 탄생된 동 서클의 이름인 다솜은 사랑의 순수한 우리말이라고.

대체로 중학교 1학년에서 고등학교 3학년까지의 학생들로 구성된 다솜은 컴퓨터를 일찍부터 접할 수 있어 컴퓨터에 관해서는 누구보다도 많은 지식을 얻을 수 있다고 한다.

현재 회원은 11명으로 아직은 정기적인 모임을 가지고 있지 않지만 통신을 통하여 자신들의 세대가 가질 수 있는 느낌에 대하여 대화를 할 수 있어 매우 많은 도움을 받고 있다고.

동 모임은 회장이나 특별한 임원진을 구성하지 않은 것이 그 특징이다. 앞으로도 계속 청소년의 신분을 지키면서 컴퓨터를 사랑하는 사람으로 남기를 원한다고 한다.

다솜에 가입하려면 H-Mail 사용자로 고등학교 3학년 이하의 사람이면 누구라도 가입할 수 있다.

다솜에 관한 문의 사항은 전화번호 (02)336—4585 고연희씨 앞으로 하면 된다.

●씨알
진실된 젊은이들의 모임

씨알은 H-mail 사용자중에 20대 초반의 젊은이들이 모여 서로가 같은 공감대를 가지고 서로의 이야기를 진솔하게 나누고자 하여 지난 88년 9월에 몇몇의 젊은이들이 모여 탄생시킨 모임이다.

처음에는 5명으로 시작하였지만 현재는 32명의 회원을 확보하고 있으며, 이들은 매달 두번째주 토요일에 정기적인 모임을 가지고 있다. 이 모임에서는 주제를 선정하여 토론회 형식으로 진행하고 있는데, 주제는 전산 분야에 국한하지 않고 문학, 예술, 철학 등 젊은이들로서 알아야 할 다양한 분야에 걸친 것들을 다룬다. 그리고 지난달 10일부터 20일까지는 H-Mail 사용자중에 컴퓨터에 익숙하지 않은 사용자들을 위하여 기초적인 컴퓨터사용에 대하여 교육을 실시하기도 하였다.

씨알에 가입하려면 H-Mail 사용자로 고등학교를 졸업하고 25세 이하인 사람으로 관심을 가진 사람은 누구나 가입할 수 있으며 회비는 모임이 있을 때마다 임시적으로 수령하고 있다.

김보현 회장(경원대 전산과2)은 앞으로는 이야기 위주의 방식을 탈피하고 서로 다른 분야의 정보교환을 위한 모임으로 확대시킬 계획이라고 한다.

씨알에 관하여 알고 싶은 분은 전화번호 (02)336—4585 고연희씨 앞으로 하면 된다.

●약사 컴퓨터 동우회
전국민 의료보험제도에 맞는 프로그램 개발예정

약사 컴퓨터 동우회는 고객에게 양질의 정보를 제공하고, 업무처리 및 고객관리 등에 신속함을 꾀하며, 약국의 컴퓨터 마인드를 확산시켜 회원간의 친목도모를 위해 85년 1월 약사 10명이 모여 조직하였다.

동 모임은 매년 창립 기념일에 정기총회를 통하여 그동안 개발되었던 프로그램의 강좌 및 발표회를 가지고 있으며, 발표된 의견들은 상임위원회에서 처리하고 있다. 또한 약국관리 프로그램을 도입하여 조제록관리, 조제상담, 한방과립제, 고객 체질감별, 질병에 따른 식단표 등 여러 분야에 걸쳐 약국업무의 효율화를 도모하고 있다.

현재 전국적으로 236명의 회원을 확보하고 있는 동 모임은 회원들의 교육은 교육담당 요원이 회원들을 방문하여 직접 해결하고 있다.

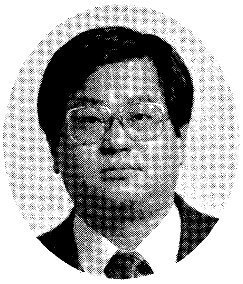
동 모임에 가입하려면 컴퓨터에 관심을 가진 대한 약사회의 회원으로서 프로그램 개발과 교육에 필요한 일정액의 비용을 납입하면 된다.

이광구 회장(수원 보경당 약국)은 앞으로의 계획으로 금년 7월에 시행될 전국민 의료보험제도에 발맞추어 약국 실정에 맞는 프로그램을 개발하여 국민의 건강상담 및 진료, 치료 등에 이용할 수 있도록 하는 것이라고 한다. 또한 회원관리나 정기적인 모임을 모뎀을 이용한 통신을 통하여 활발히 하여 회원간에 정확하고 신속한 정보를 교환할 수 있도록 할 방침이라고 한다.

약사 컴퓨터 동우회에 관한 문의 사항은 전화번호 (0331)5—3561 이광구 회장 앞으로 하면 된다.

●엠팔

89년 상반기 임원진 선출



지난해 12월 17일 엠팔의 7차 모임이 전자까페에서 있었다. 이번 모임에서는 89년도 상반기 회장 및 총무의 선출이 있었는데 회장에는 장왕교재연구원의 상무이사인 이기성씨(사진)가, 총무로는 공영종합시스템의 박성현씨가 선출되었다.

엠팔 회장의 임기는 6개월이며 신임 회장인 이기성씨는 엠팔내의 기술적인 지원그룹인 초이스 그룹의 자문과 엠팔내의 작은 모임인 다솜, 씨알, 램당 등의 의견을 최대한으로 수렴하여 모든 일을 결정할 것이며, 현재 표준화된 2바이트 완성형 한글코드가 모든 한글을 표시할 수 없는 단점이 있기 때문에 2바이트 조합형 한글도 병행하여 사용할 수 있도록 함과 동시에, 현재 쓰이고 있는 단위나 숫자의 개념이 외국의 단위를 위주로 사용되고 있는 사실을 감안하여 년도를 표기함에 있어서 서기로 표현하기보다는 단기로 표현하는 등 한글 맞춤법 통일안에 준하여 우리 것 찾기운동을 전개할 계획이라고 한다.

신임 회장인 이기성씨의 약력은 다음과 같다.

- 서울대학교 졸업(1968년)
- PC프로그래머 협회 고문역임
- 대한출판협회 업무전산화 추진의원
- 한국전자출판연구회 이사
- 한국출판인대학 강사
- 정보문화센터 강사
- 신구전문대학 강사
- 동국대학교 정보산업대학원 강사

●연세대학교 의과대학 컴퓨터써클

가장 먼저 시작한 터줏대감

쎄컴(SECOM:Severance Computer)이란 이름으로 연세대학교 의과대학 컴퓨터써클이 탄생한 것은 지난 85년 12월이다.

부원간의 원활한 관계유지 및 컴퓨터의 프로그램기술에 대한 실력을 향상시켜 의과대학내의 컴퓨터마인드 조성이라는 목적을 가지고 25명이 주축이 되어 시작한 쎄콤은 그동안 의과대학내에서 시험을 치룰 때 OMR카드를 사용하는데, 그 카드를 처리해주는 성적합산 프로그램을 제작하였으며 또한 의대2학년의 자료들을 모은 '족보'를 제작하여 전공공부에 참고로 사용하고 있다.

쎄컴은 국내 의대 컴퓨터 써클중에서는 최초로 탄생한 써클로서 현재 회원은 60여명 정도된다. 회원 모집은 동계방학과 하계방학 그리고 학기초로 나누어 실시하고 있다.

동 써클은 86년부터 매년 1회씩 의과대학 축제기간에 회원들이 짠 프로그램과 하드웨어를 선보이는 전시회를 실시하고 있다.

또한 회원들의 교육은 신입생을 위하여 매주 1회의 컴퓨터 강습회를 실시하여 MS-DOS, 워드프로세서와 통계처리를 위한 SPSS와 같은 스프레드시트 등을 교육하고 있으며, 이 강습회는 일반 재학생에게도 개방하고 있다. 그리고 기존의 회원을 위해서는 자료정리에 도움이 되고, 업무용 프로그램으로도 유용한 디베이스 강의를 실시하고 있다.

김광원 회장(본과 2)은 동계방학을 이용하여 신입생을 위한 기초 컴퓨터교실을 개설하여 교육할 것이며 디베이스에 관하여도 집중적으로 연구할 계획이라고 한다.

쎄콤에 관한 문의 사항은 전화번호 (02)386—0325 김광원 회장 앞으로 하면 된다.

●OSIA TG4

올해안에 114안내 서비스 상용운영할 예정

국제적인 통신 표준화 기구(ISO)와 국제 전신 전화 자문위원회(CCITT)의 권고사항인 OSI(Open System Interconnection)가 국가기간 전산망 정책의 실시로 국내에서도 실용화될 것을 예상하고 그에 대한 실시를 목적으로 지난 87년 4월에 OSIA라는 전문 연구기관이 설립 되었지만 PC통신 서비스 및 제어절차 즉, 프로토콜을 담당하는 기술 그룹의 필요성이 대두되자 OSIA TG4가 지난해 11월 24일 첫 모임을 가지게 되었다.

현재 회장에는 강철희 박사(한국 전자통신연구소 정보통신 부장)가, 간사에는 정해원씨가 맡고 있으며 모임은 비정기적으로 격월로 가지고 있다. 모임에서는 프로토콜에 관련된 주제를 선정하여 발표회를 갖고 의견을 나누는 형식으로 진행하고 있다.

회원에 가입하려면 산업계를 중심으로 PC통신의 표준화 작업에 적극적으로 참여하고자 하는 사람으로 1년에 만원의 회비를 납부하면 된다.

TG4는 조속한 시일내에 국내에 널리 상용될 전자우편 및 사서함, 데이터베이스 검색 및 MHS(Message Handling System) 서비스 등에 대하여 산업계를 중심으로 표준화작업을 실시할 예정이며, 89년도에는 114안내 DB 검색서비스와 MHS서비스를 표준화하여 내년부터 상용으로 운영하도록 할 방침이라고 한다.

또한, 앞으로의 장기적인 계획으로는 해외의 PC통신망 및 서비스를 소개하고, 관련된 국제 표준의 동향을 분석하는 등 국내의 PC통신 활성화를 위한 기반을 조성할 계획이라고 한다.

OSIA TG4에 관한 문의 사항은 전화번호 (042)820—6154 정해원씨 앞으로 하면 된다.

●유니코사

89년 신임회장단 선출

유니코사(전국 대학 컴퓨터 써클 연합회)는 지난해 12월 정기 총회에서 89년도를 이끌어갈 신임회장으로 최재흠(한양대(안산) 전산과 2)씨를 선출했다.

최재흠씨는 89년도 유니코사의 운영계획에 관하여 우선 회지와 학술지, 교육지 등을 발행하여 유니코사내의 학술활동을 활발히 전개할 것이며 현재 경인지역별로 활동하고 있는 대학 컴퓨터 써클을 충청지구, 대구지역, 부산지역, 전남지역 등의 지방 연합회와 작품 및 학술 교류를 통하여 총 연합 전시회를 12월에 개최하여 총체적인 연합회를 구성하는 것이라고 밝혔다.

또한 유니코사에서 자체적인 적립금 제도를 마련하여 자체 활동을 위한 경제적인 비용을 마련해 갈 것이며, 전체 유니코사회원이 증원되면서 연합회의 성격이 흐려지자 회원들의 연합회에 대한 적극성을 고양시키기 위해 사무실을 이용한 세미나나 교육 등을 장려할 것이라고 한다.

현재 유니코사는 방학을 이용한 집행위원들의 사업수립을 위한 모임이 매주 월요일과 목요일에 있으며, 지난 1월 20일부터 23일까지 집행위원들의 MT가 있었다.

●한국어 전산학회

회보 '새소식' 창간하고 정보은행 설치할 예정

한국어 자료 및 정보를 수집, 정리, 전산화하고 그 효율적인 활용 방안을 강구하여, 한국어의 연구와 보급을 증진한다는 목적으로 지난 88년 2월 27일에 발족한 한국어 전산학회는 현재 81명의 회원으로 구성되어 있다.

동 학회는 연 6회의 연구 발표회를 통하여 어문학연구를 위한 개인용 컴퓨터의 활용, 고어 폰트자의 통일, 한글-한자 단어 변환 프로그램의 문제 해결 구조와 그 활용 등 어문학 특히 국어학과 관련된 분야에서의 컴퓨터의 이용에 관한 주제를 발표하여 왔으며, 매월 1회의 활용강좌를 실시하여 컴퓨터의 구체적인 활용에 관하여 회원들의 이해를 돕고 있다.

그리고 지난해 12월에는 회보인 '새소식'을 창간하여 회원들의 활동사항을 알리고 있다.

임원진은 회장에 이기문(서울대 국문과)교수, 부회장에 김충회(인하대 국문과)교수, 김홍규(고려대 국문과)교수, 감사에 남풍현(단국대 국문과)교수, 이현복(서울대 언어학과)교수, 총무이사에 박성종(관동대 국어교육과)교수, 연구이사에 김정수(한양대 국문과)교수, 정인상(충북대 국문과)교수, 출판이사에 홍윤표(단국대 국문과)교수, 지방이사에 전북지역의 김병선(전북대 국문과) 교수가 맡고 있다.

동 학회는 단기적인 사업계획으로 한국어 관계 데이터 베이스를 구축하기 위한 입력구조의 통일과 문헌 목록과자료의 정리를 계획하고 있으며, 또한 장기적인 사업으로 연구 논문집인 회지의 간행과 한국어문학 관계 자료집의 간행과 자료 화일의 보급 및 정보은행의 설치에 관하여 계획하고 있다.

동 학회에 가입하려면 우선 컴퓨터를 이용하여 국어에 관련된 일을 하는 연구자로서 정회원 2인 이상의 추천을 받아 가입비 만원과 함께 입회원서를 임원회에 제출하여 승인을 받아야 한다.

동 학회에 관한 문의 사항은 전화번호 (02)734—9860, 9866 한국어 전산학회 앞으로 하면 된다.

●한양 컴퓨터연구회

하드웨어와 소프트웨어의 고른 실력을 배양

한양대학교 안산캠퍼스 학생들로 구성된 한양 컴퓨터연구회(하이코라:HY-CoRA)는 젊은이로서 무엇인가를 이루겠다는 신념을 바탕으로 컴퓨터 마인드의 확산을 위한다는 목적으로 창립된 써클이다.

지난 84년 9월 창립되어 현재 회원이 80명에 이르고 있는 하이코라는 신입회원들을 매학기초에 공개 모집하고 있다.

하이코라는 매년 1회의 초청 강연회를 실시하여 외부인사를 초청하여 자신들의 나아갈 방향이나 컴퓨터에 관한 지식을 배워가고 있다. 그리고 각각 서로 다른 수준인 회원들의 실력향상을 위하여 7명이 한개의 조를 이루어 한주에 3회정도 실시하는 조별 교육과 1주일에 1회의 컴퓨터 기초교육과 워드프로세서 교육을 실시하여 컴퓨터를 처음 접하는 회원들의 이해를 돕고 있다. 특히 매년 개최하고 있는 전시회는 학기중과 방학중에 회원들이 만들어 낸 작품을 선보이고 있는데 타학교의 써클에 비하여 소프트웨어와 하드웨어를 고르게 전시하고 있다.

송원식 회장(기계과 2)은 현재 하이코라의 활동에는 동계방학을 이용하여 스터디를 조직하여 파스칼이나 C 등의 프로그램언어에 대한 집중적인 연구가 있으며, 89년도 계획으로 타 학교의 써클과 교류를 원활히 하고 회원증원에 따른 써클의 단합심을 강화하며 자체 실력을 향상시키기 위해 조별교육을 강화할 것이라고 한다.

알려드립니다

본지에서는 써클 동정을 통해 전국에 산재되어 있는 아마추어들의 모임을 소개하여 동호인끼리 컴퓨터 이용자 모임을 형성할 수 있도록 유도하고 있읍니다.

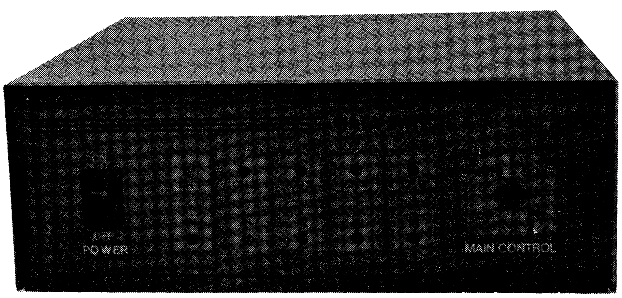
이용자 모임을 알리고자 하는 분은 본지로 연락해 주시기 바랍니다.

연 락 처 : 서울시 용산구 갈월동 17번지 세기빌딩 ㊞ 140-150
전화번호 : (02)754—1044
F A X : (02)757—1049

PRINTER 주변기기 전문 메이커

- ●컴퓨터는 2대 혹은 3대이고, 프린터는 1대뿐이라 불편하게 사용하고 계십니까?
- ●컴퓨터 1대에 프린터를 1대씩 사용하거나 추가 구입시 프린터를 구매하려고 생각하십니까?
- ●프린터를 하고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- ●CAD 사용시 Plotter가 그리고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- ●컴퓨터의 수량과 프린터의 수량이 안맞아 적절한 사용을 못하고 계십니까?
- ●학교 및 학원의 실습실에 프린터대수가 부족해 적절한 교육을 못하고 계십니까?
- ●F.A 혹은 계측기쪽의 프린터 인터페이스에 문제가 있읍니까?

컴퓨터 사용중에 불편한 위의 모든 문제를 자신있게 처리해 드리므로, 귀사의 업무에 도움이 되어 드리고 싶읍니다!

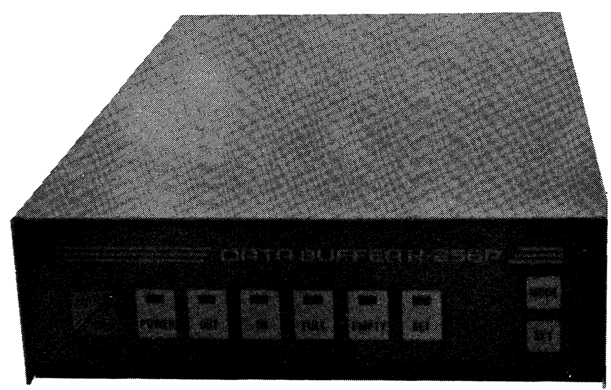


K-P-345A의 사양

입·출력 방식	Centronics Type
입력 COMPUTER	3:1~5:1
출력 PRINTER	1대
대기 시간	5초, 10초, 20초, 40초 중 선택
채널 변환방식	전자식
채널 선택방식	자동 or 반자동
용지 F.F 기능	자동 F.F or None F.F
F.F 삭제 기능	자동 삭제 or None

K-256P의 사양

버퍼 용량	256kB(80col 약 200장)
입·출력 방식	Centronics Type
입·출력 수	1:1 or 2:1
사용자 MODE	10EA
고유 기능	Data 중 Space 입력 기능, 재 Copy 기능, 메모리 기능, Self Test 기능 등 총 10가지



● DATA SWITCH

- 전자식 완전자동 (2:1~8:1 or 64:1)
- 전자식 반자동 (2:1~8:1, 4:2)
- Serial 완전자동(4:1, 8:1)

● DATA BUFFER

- 소용량 64kB
- 고성능 다기능 256kB
- 대용량 다기능 1MB
- Serial용 1MB

● DATA CONVERTOR

- Serial→PARALLEL
- PARALLEL→Serial

● 각급 학교 및 학원의 전산 실습실 프린터 공유장치 공사

ATTA SYSTEM
PRINTER 주변기기 전문메이커
서울·종로구 종로 3가 175-4(현대APT709호)
전 화:277-1739

●상기 제품은 아래의 전문 취급점에서 전시 판매하고 있으며 직접 사용해 보실 수 있읍니다.●

서울 : 세운상가 협진 274—5419(가동 라열 426호)
용산 나진상가 동해전자 719—8010(17동 가열 304호)
용산 전자랜드 협진 712—2764(3층 C-336호)

인천 : 바이너 리컴퓨터 (032)762-5544
대구 : 범진 컴퓨터 (053)252-2211
마산 : 한길 컴퓨터 (0551)94-5555
원주 : 중부 시스템 (0371)43-7979

(기타 지역 전문 취급점 모집중)

조합논리회로

김정선/항공대 전자과 교수

디지탈회로는 크게 두 부류로 나뉘어 진다. 즉 전의 입력조합에 관계없이 현재 입력들의 조합에 의해 출력이 직접 결정되는 조합논리회로(Combinational Logic Circuit)와 현재입력과 현재의 출력상태에 의해 다음의 출력상태가 결정되는 순서논리회로(Sequential Logic Circuit)이다. 순서논리회로는 다음에 다루기로 하고 우선 조합논리회로를 다루기로 한다.

조합논리회로는 부울함수(Boolean Function)에 기인하여 논리적으로 규정된 동작을 수행하는 회로로서 그림 1과 같은 블록도로 표현된다.

여기서, X는 입력변수(독립변수)이고 Y는 출력변수(종속변수)이다. 따라서 임의의 시간에서 출력Y는 입력X에 의해서만 결정되는 함수의 형태로 표현된다.

즉 출력Y의 함수는

$$Y = f(X_1,\ X_2, \cdots X_n)$$

로 표현된다. 이 함수의 각 변수는 2진 변수(변수값이 0과 1만 가능)이므로 조합가능한 입력의 경우의 수는 2^n개가 된다. 그리고 각 출력에 따라 출력의 응답형태를 결정하는 논리회로의 경우의 수는 2^k(단, $k=2^n$)이 된다.

또한 입력변수를 공유하는 단일 출력 네트워크의 조합은 멀티 출력 네트워크를 구성한다.

멀티 출력 네트워크는 그림 1(b)와 같고 함수의 형태로 표현하면 다음과 같다.

$$\begin{aligned} Y_1 &= f_1(X_1,\ X_2, \cdots X_n) \\ Y_2 &= f_2(X_1,\ X_2, \cdots X_n) \\ &\ \vdots \\ Y_P &= f_P(X_1,\ X_2, \cdots X_n) \end{aligned}$$

따라서 멀티 출력일 경우 논리회로의 경우의 수는 $(2^k)^p$(단, $k=2^n$)이 된다.

〔표 1〕 진리표

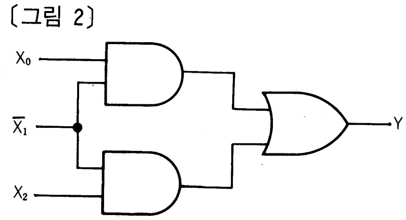
X_2	X_1	X_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

조합회로의 설계과정

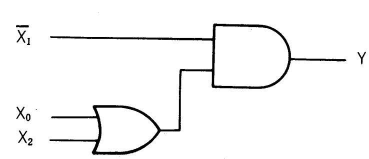
조합회로의 설계는 문제에 대한 대략적인 기술에서 시작하여 논리회로도를 쉽게 얻을 수 있도록 부울함수를 조합함으로써 끝난다. 그러면 임의의 원하는 기능을 갖는 조합회로를 설계할 경우 그 과정을 살펴보자.

① 조합회로로 구성할 수 있는 문제를 설정한다.
② 필요한 입력과 출력변수의 수를 결정한다.(n, p)
③ 입 · 출력 관계를 나타내는 진리표를 작성한다.
④ 진리표로부터 최소화된 출력의 부울함수를 얻는다.
⑤ 조합회로의 논리선도를 작성한다.
⑥ 게이트수, 팬아웃, 지연시간, 게이트의 구동용량 등을 고려하여 수정여부를 검사한다.

〔그림 2〕

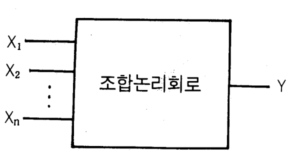


(a) $Y = X_1' X_0 + X_2 X_1'$의 논리선도

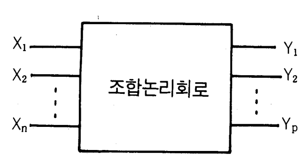


(b) 간략화된 논리선도

〔그림 1〕 조합논리회로



(a) 단일 출력 네트워크



(b) 멀티 출력 네트워크

이와 같은 설계과정을 토대로 간단한 예를 들어 보자. 입력변수 세개와 출력변수 하나로 각 입·출력관계를 나타내는 진리표가 표 1과 같다면, 출력 Y의 함수는

$$Y = X_2'X_1'X_0 + X_2X_1'X_0' + X_2X_1'X_0$$

이다. 부울대수를 사용하여 간단히 하면,

$$Y = X_1'X_0 + X_2X_1'$$

이다. 이식을 논리선도로 나타내면 **그림 2** (a)와 같이 된다. 그러나 위의 출력함수 Y는 다음과 같이 더욱 간략하게 할 수 있다. 즉 $Y = X_1'(X_0 + X_2)$로 이를 논리선도로 나타낸 것이 그림 2 (b)이다. 여기서 (a)보다 (b)가 게이트수가 적은 것을 볼 수 있다.

그러면 일반적으로 널리 사용되고 있는 기본 조합논리회로인 가산기(Adder), 감산기(Subtracter), 곱셈기(Multiplier), 비교기(Comparator), 디코더(Decoder), 멀티플렉서(Multiplexer : Mux or Selector) 및 인코더(Encoder) 등에 대해 알아본다. 그러나 기본적인 조합논리회로로서 배타적 논리합(Exclusive – OR)과 표시장치(Display Device), 코드변환기(Code Converter) 및 패리티 체커(Parity Checker) 등도 있다.

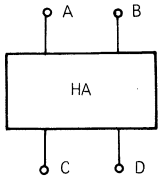
가산기

그림 3(a)와 같이 입력 A와 B를 더한 합인 출력 D(Sum : 합)와 출력 C (Carry :캐리)를 얻는 회로를 반가산기(Half Adder)라 한다. 즉 입력으로 2개의 2진 숫자를 받아들이고 그 출력으로 2개의 2진 숫자인 D(Sum)와 C(Carry)를 만들어 내는 것이 반가산기이다. 그 진리표는 그림 3(b)와 같다.

각각 2개의 2진 출력의 논리식은

$$D = A'B + AB' = A \oplus B$$
$$C = AB$$

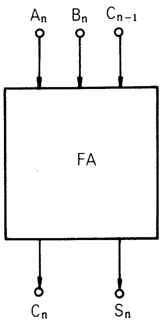
〔그림 3〕 반가산기(HA)와 전가산기(FA)



(a) 반가산기

입	력	출	력
A	B	C	D
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) 반가산기의 진리표

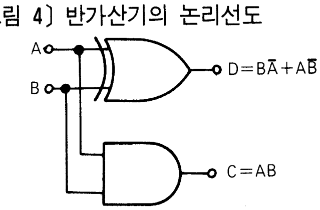


(c) 전가산기

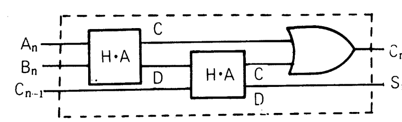
선	입	력		출	력
	a_n	B_n	C_{n-1}	S_n	C_n
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

(d) 전가산기의 진리표

〔그림 4〕 반가산기의 논리선도



〔그림 5〕 반가산기 두개로 구성된 전가산기



와 같이 된다.

이 논리식에서 반가산기는 배타적논리합(Exclusive–OR)과 AND게이트로 구성됨을 알 수 있다. 이 논리식에 의해 **그림 4**에 반가산기의 논리선도를 나타냈다.

두 자리 이상의 2진수를 더하는 경우 2개의 같은 자리수들끼리 동시에 더하는 것을 평행연산이라 하고 이를 수행하기 위해서는 앞단의 자리올림수(Carry)까지 합해서 처리할 수 있어야 한다. 따라서 그림 3(c)에서와 같이 $A_n + B_n + C_{n-1}$의 세개의 입력을 더할 수 있어야 한다. 이를 전가산기(Full Adder)라고 한다. 그림 3(c)에서 볼 수 있듯이 전가산기는 3개의 입력을 받아들여 합인 S(Sum)와 캐리(Carry)를 만들어 내는데 이 3개의 입력중 하나가 바로 앞단의 캐리인 것이 반가산기와의 차이점이다. 이러한 전가산기는 반가산기 두개로 구성할 수 있다. 바로 **그림 5**가 두개의 반가산기로 구성된 전가산기이다.

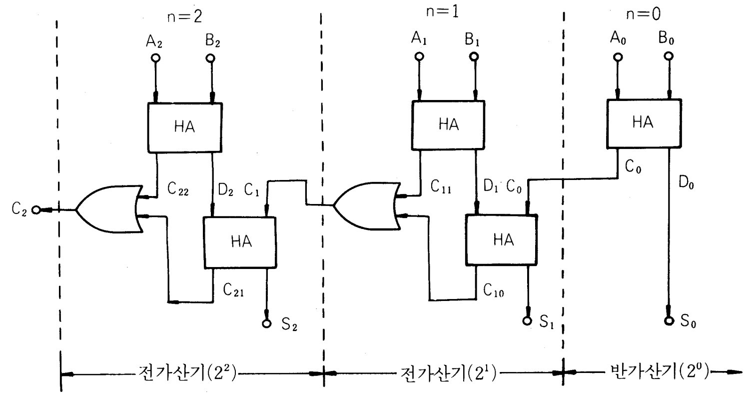
반가산기로 구성된 평행 2진가산기 회로를 **그림 6**에 보인다. 첫단은 그 앞단의 자리 올림수가 없으므로 반가산기를 사용한다. 따라서 첫단을 제외하면 모두 같은 모듈의 전가산기가 반복된다.

그림 6에서 전가산기를 반가산기를 이용해서 구성한 것을 보였는데 이제 그림 3의 (c)(d)를 참고하여 직접 설계해 보자.

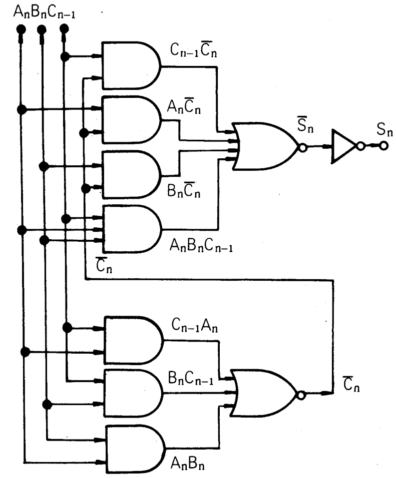
우선 출력함수를 보면

$$S_n = A_n'B_n'C_{n-1} + A_nB_n'C_{n-1}' + A_nB_n'C_{n-1}' + A_nB_nC_{n-1}$$
$$C_n = A_n'B_nC_{n-1} + A_nB_n'C_{n-1} + A_nB_nC_{n-1}' + A_nB_nC_{n-1}$$

〔그림 6〕 반가산기로 구성된 평행 2진가산기



〔그림 7〕 전가산기 논리선도



와 같다. 이를 간소화하면,

$$C_n = B_nC_{n-1} + C_{n-1}A_n + A_nB_n$$
$$S_n = C_{n-1}C_n' + B_nC_n' + A_nC_n' + A_nB_nC_{n-1}$$

이를 AOI(AND–OR–INVERTER) 회로로 구성하면 그림 7과 같다. 이들을 연속해서 사용할 때는 물론 앞단 캐리와 연결하면 얼마든지 확장할 수 있다. 2단 또는 4단 전가산기는 MSI(중규모 집적회로)로 제작되어 나온다. 입력A와 B가 동기화된 펄스의 연속으로 진행할 때 이를 순서적으로 합하여 내는 것을 직렬연산(Serial Operation)이라 하고, 그림 8과 같이 가산을 직렬 연산으로 수행한다.

그림에서 TD회로는 한스탭 시간을 지연하는 회로로 Time Delay의 약자이다. 이는 평행연산 가산보다 시간은 오래 걸리나 자리수를 무한히 확장하여 연산할 수 있는 장점이 있다.

감산기

감산은 한 입력을 보수(Complement)로 하여 가산기에 가하면 가능하지만 감산기를 직접 설계해 보자.

A–B를 수행하는 경우 차(Difference) 출력과 자리내림(Borrow)출력이 요구되고 진리표는 표 2와 같다.

출력함수는

$$d(차) = A'B + AB' = A \oplus B$$
$$b(자리내림) = A'B$$

와 같고 이를 논리선도로 그리면 그림 9와 같다.

전감산기는 i번째 자리의 감산인 경우 $A_i - B_i$가 되고 이에 해당하는 진리표는 표 3과 같으며 b_i는 i번째 자리의 자리 내림수이다.

따라서 전감산기의 함수는

$$b_{i+1} = A_i'B_i'b_i + A_i'B_ib_i' + + A_i'B_ib_i + A_iB_ib_i = A_i'B_i + A_i'b_i + B_ib_i$$
$$d_i = A_i'B_i'b_i + A_i'B_ib_i' + A_iB_i'b_i' + A_iB_ib_i$$

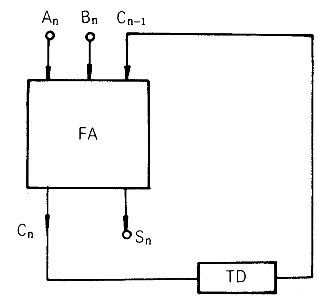
이고, 이의 논리회로는 그림 10과 같다.

이것도 마찬가지로 반감산기 두개를 사용하여 전감산기를 구성할 수 있는데 그림 11과 같이 나타낼 수 있다. 이제까지 가산기와 감산기에 대해 설명하였다.

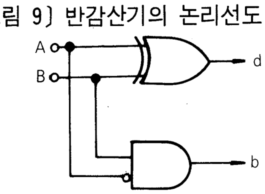
진보영1기

다음은 진보영1기(眞補零1器 : True–Complement–Zero–One)에 대하여 알아 보자. 가산기와 감산기를 같이 쓰기 위하여 보수로 하기 위한 제어기가 필요하다. 이 제어기가 바로 진보영1기 이다.

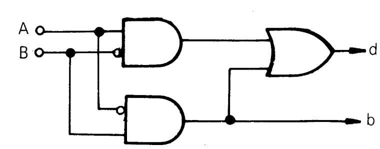
〔그림 8〕 직렬 2진 전가산기



〔그림 9〕 반감산기의 논리선도



(a)



(b)

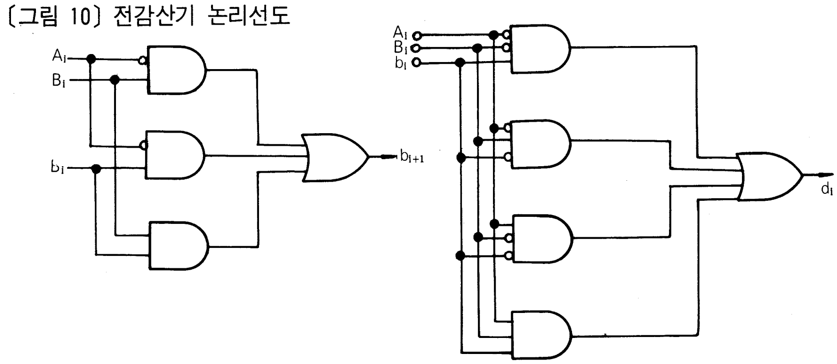
〔표 2〕 반감산기의 진리표

입력		출력	
A	B	자리내림(b)	차(d)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

〔표 3〕 전감산기의 진리표

입		력	출	력
A_i	B_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

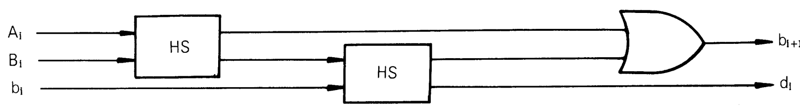
〔그림 10〕 전감산기 논리선도



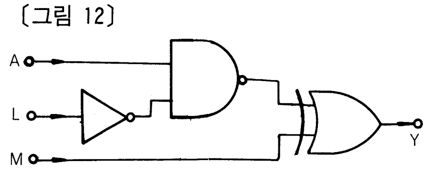
〔표 4〕 비교기의 진리표

입	력	출	력	
A_n	B_n	C(A>B)	E(A=B)	D(A<B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

〔그림 11〕 두개의 반가산기로 된 전가산기



〔그림 12〕



(a) 진보영1기

입	력	출 력
L	M	Y
0	0	$\overline{A}$
0	1	A
1	0	1
1	1	0

(b) 진리표

〔그림 13〕 4단 가감산기

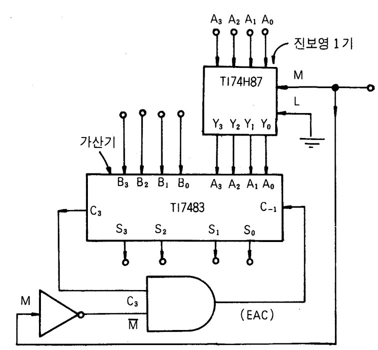


그림 12에서 입력 L과 M은 제어를 위한 입력이고 A는 데이터를 의미한다. L이 0이고 M이 0일때는 출력Y에는 $\overline{A}$가 나타나나 M이 1일때는 출력 Y에 $\overline{A}$의 보수인 A가 나타난다. 그에 대한 진리표는 그림 12(b)에서 볼 수 있다.

이제 우리는 진보영1기를 이용하여 가감산을 동시에 할 수 있게 되었다. 진보영1기는 실제회로에 많이 사용되며 4개의 이 회로를 갖는 MSI도 있다. 7487 TTL(Transistor Transistor Logic) 이 그 예이다. 4단 가산기와 이 제어소자를 연결하여 가감산을 할 수 있는 회로를 **그림 13**에 보였다. 이때 가산기는 어느 4단가산기라도 좋으나 여기서는 7487 TTL IC를 사용했다.

이를 가산기로 쓸 경우 그림 13에서와 같이 L은 접지되어 0이 되어 있으므로 M을 1로 하면 Y_n이 A_n으로 가산기가 된다. M을 0로 하면 Y_n은 $\overline{A_n}$로 되어 보수(Complementary)가 되므로 감산기로 동작한다.

여기서 EAC는 최종올림수(End Arround Carry)로서 초단에 가해서 결과를 얻는다.

에를 들어 1101-1000을 행할 경우

1101 ········· $B_3B_2B_1B_0$
+0111 ········· $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
①0100
+ 1 ········· EAC (C_3)
0101 ········· $S_3S_2S_1S_0$

EAC가 최종적으로 합해져 결과적으로 0101을 얻는다.

비교기

입력A와 B를 비교하여 A=B, A〉B, A〈B의 출력을 얻는 것을 2진 비교기라 한다. 이의 진리표를 **표 4**에 보였다.

이에 의해 출력식은

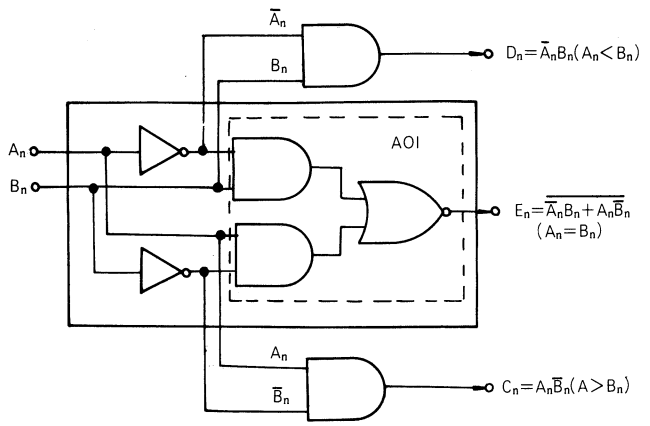
$$E=(A'_nB_n+A_nB'_n)'$$
$$C=A_nB'_n$$
$$D=A'_nB_n$$

로 표현된다.

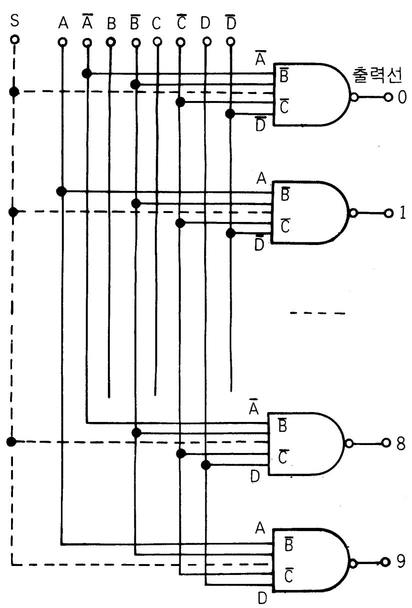
윗식으로부터 **그림 14**와 같이 논리회로를 구성할 수 있다.

그림 10은 n번째의 한자리만 비교하는 회로이나 이를 기초로 여러가지 비교기도 만들 수 있다.

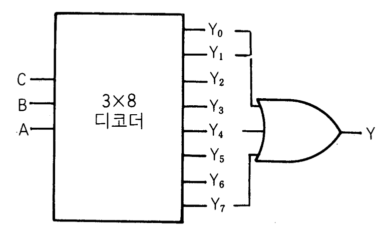
〔그림 14〕 한자리 2진 비교기



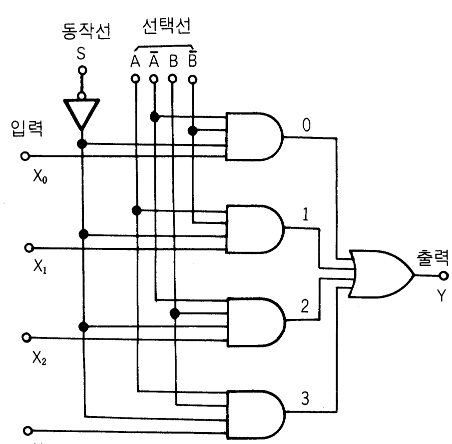
〔그림 15〕 BCD의 10진 디코더



〔그림 16〕 3×8 디코더



〔그림 18〕 멀티플렉서



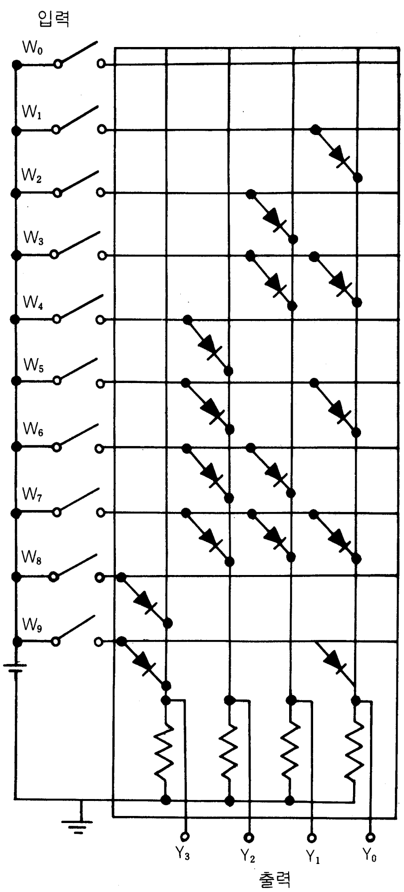
〔표 5〕 디코더의 진리표

C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

〔그림 17〕 인코더를 위한 진리표와 회로

입력	출		력	
W_n	Y_3	Y_2	Y_1	Y_0
W_0	0	0	0	0
W_1	0	0	0	1
W_2	0	0	1	0
W_3	0	0	1	1
W_4	0	1	0	0
W_5	0	1	0	1
W_6	0	1	1	0
W_7	0	1	1	1
W_8	1	0	0	0
W_9	1	0	0	1

(a) 진리표



(b) 회로

디코더

입력조합에 따라 출력이 하나만 동작하도록 하는 것을 디코딩(Decoding)이라고 한다. 가령 3비트로 된 입력은 8가지의 다른 조합을 이룰 수 있고 출력이 8개로 이루어져 한 조합에 대하여 한 출력만 동작하게 할 수 있다. 이를 진리표로 나타내면 표 5와 같다.

BCD 10진 디코더는 4개로 된 BCD 코드를 4입력 AND회로에 연결하여 AND의 출력이 1이 되도록 동작시킨

다. BCD 코드의 디코더는 그림 15에 나타낸다. 여기서 S입력은 이 디코더의 인에이블 단자로 동작한다.

또한 디코더의 출력은 각각 민텀(Minterm) 을 의미하므로 앞의 진리표인 표 5를 디코더를 이용하여 구현할 수 있는데 곱항의 합으로서 OR 게이트를 사용하면 그림 16과 같이 된다.

인코더

디코더의 반대역할을 하는 것이 인코더이다. 인코더는 여러개의 입력과 출력을 갖춘 회로에서 임의의 한 입력단자에 신호가 가해졌을때 그 입력단자에 대응하는 출력단자의 조합에 신호가 N비트 코드로 발생되는 장치이다.

즉 입력이 10개이고 그중 어느 하나가 1이 될때 출력이 $Y_3Y_2Y_1Y_0$의 코드로 되고 그림 17(a)와 같은 진리표에 의한 것을 얻고자 할 때는 특정입력 W_n이 1일 때 원하는 출력의 조합을 얻어야 하므로, 입력선과 출력선의 행렬을 그림 17(b)와 같이 형성하여 입력이 1인 횡선과 출력이 1이 되는 종선사이에 모두 다이오드를 순방향으로 연결하면 된다.

멀티플렉서

멀티플렉서는 N개의 입력데이터에서 1개 입력씩 선택하여 단일 채널을 통해 전송하는 것을 말한다. 즉 4개의 입력이 X_0, X_1, X_2, X_3일때 멀티플렉서는 그림 18과 같다. 이때 A와 B로 전송할 입력 X_0~X_3중 하나를 선택하여 출력에 전송한다. PC

D/A 콘버터

최승욱 /태성전산

우리가 일반적으로 컴퓨터의 인터페이스를 떠올릴 경우 대표적인 것으로 A/D 콘버터와 D/A 콘버터를 들 수 있다. 물론 컴퓨터의 인터페이스는 반드시 아날로그 시스템과 디지탈 시스템만이 연결되는 것은 아니지만 실제 외부 제어대상의 측정량들이 대부분 아날로그적인 연속적 양이기 때문에 이와 같은 생각을 하는 것이다. A/D 콘버터와 유사하게 D/A 콘버터도 여러종류의 것이 있고 어떤 의미에서는 D/A 콘버터가 A/D 콘버터보다는 먼저라는 인상도 있다.

여러분이 A/D 콘버터의 내용을 상기할 때, A/D 콘버터중 계수형 A/D 콘버터는 외부의 입력값 V_x를 자체내에서 발생하는 D/A 콘버터의 출력과 비교하여 그 값을 얻는 것을 떠올릴 수 있을 것이다.

이와 같이 D/A 콘버터는 실제로 A/D와 D/A 콘버터를 모두 만족하는 성질을 갖고 있다고 볼 수 있다. 그러면 여기서는 D/A 콘버터의 종류와 각각의 특성, 선택요령과 제작 등에 관하여 설명하겠다.

D/A 콘버터의 종류

D/A 콘버터는 여러종류가 있으며 대략 다음과 같다.

1. R–2R 래더방식
2. 가산기방식
3. 직렬스위치 이득가변방식
4. 병렬스위치 이득가변방식
5. 정전류원 스위칭방식
6. 펄스적분방식

1. R–2R 래더방식

일반적으로 가장 간단한 형식으로 D/A 콘버터를 구성하려는 경우에 비교적 많이 사용되는 방식중의 하나가 바로 R–2R 래더(Ladder) 방식이라고 할 수 있다. R–2R 래더방식은 제작상 성가신점이 있기는 하지만 소자선정만 올바르게 하면 무리없이 동작하는 잇점을 갖고 있다. R–2R 래더방식은 글자 그대로 저항R과 2배의 값을 갖는 저항2R을 사다리와 같은 형식으로 조합하여 구성한 회로를 지칭하여 붙여진 이름이다.

그림 1은 R–2R 래더방식을 가리킨다. 그림 1에서 각 사다리의 계단에 붙여진 저항은 접지나 기준전압과 연결된 스위치박스를 갖는데 디지탈 코드의 입력은 스위치박스의 각 스위치를 ON/OFF하여 회로의 전류분포를 변하게 한다. 이때 각 계단의 전류분포는 $\frac{1}{2}I$, $\frac{1}{4}I$, $\frac{1}{8}I$ 등의 형식으로 분포하게 된다. 만일 출력의 최대전압을 구한다면, 즉 입력된 디지탈코드가 모두 1이면(여기서는 1일때 V_{ref}와 접속된다고 생각한다.)

$$V_{max}=V_{ref}\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}\cdots\frac{1}{2^n}\right)[V]$$

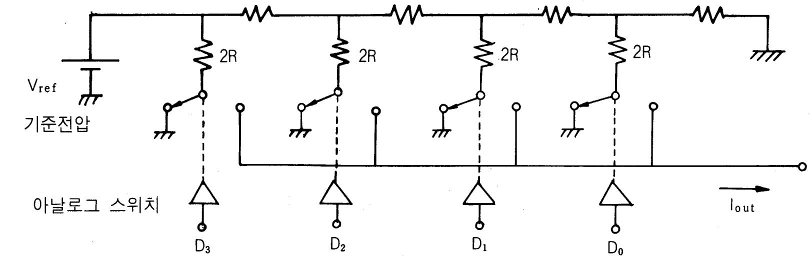
가 된다. 이때 V_{ref}는 기준전압이고 n은 입력되는 디지탈코드의 비트수가 된다. 만일 4비트 D/A 콘버터이면 n=3이 된다. 그런데, 만일 입력되는 디지탈코드가 위와같이 스트레이트 바이너리(Straight Binary)가 아니고 BCD 코드이면 그림 2와 같은 구성이 되어야 한다.

즉, BCD 코드의 각 자리마다 R–2R 래더블록을 설정하고 이를 T형 어테뉴에이터(Attenuator)로 합성해야 한다. 만일 10진, 즉 BCD로 m자리인 경우의 최대 출력을 생각해 보면

$$V_{max}=\frac{V_R}{13}\left(\frac{9}{16}\right)\left(1+\frac{1}{10}+\cdots+\frac{13}{10^{m-1}}\right)[V]$$

가 된다.

〔그림 1〕 R-2R 래더방식 DAC



전압출력의 경우는 이와같이 T형 어테뉴에이터를 사용하여야만 되나 출력의 형식이 전류형인 경우는 성가신 T형감쇄기를 삭제하여도 된다.

개방출력전압은 각 자리마다 모두 같기때문에 BCD 자리마다 1/10씩 전류를 감소시키기 위해 각 DIGIT마다 직렬로 저항을 삽입하기만 하면 된다.

2. 가산기방식

가산기방식의 DAC(DAC는 D/A 콘버터를 가리킴)는 구성상 R-2R 래더방식과 매우 유사한 모습을 하고 유사한 동작을 한다. R-2R 래더방식의 경우 사다리의 각 스테이지에 있어서 종단에 출력구를 갖고 중간에서 전류를 빼는 형식이라면, 이는 입력에서 한꺼번에 각 무게치(Weight)에 해당하는 전류값을 받아 이를 출력하는 형식이라 할 수 있다.

여러분이 OP-Amp를 입문할 때 가장 많이 보는 것 중의 하나가 가산기회로인데 가산기방식의 DAC는 이를 그대로 사용하고 이때 저항의 묶음을 중첩형 저항트리라고 한다.

이때의 출력전류는

$$I_o=V_{ref}\left(\frac{1}{R_1}+\frac{1}{R_2}+\cdots\frac{1}{R_{n-1}}\right)[A]$$

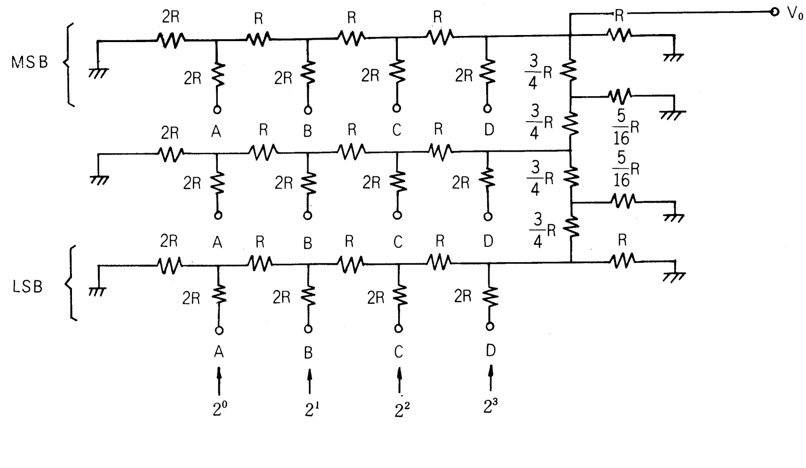
이고, $R_2=\frac{1}{2}R_1$, $R_3=\frac{1}{2}R_2$ 등의 관계가 있고 n은 비트수이다(그림 3, 4 참고).

3. 직·병렬 가변이득방식

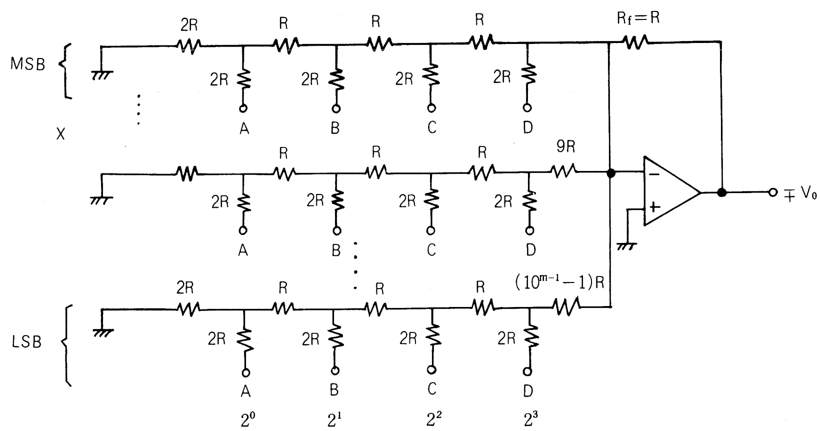
직렬스위치 가변이득방식은 일반적으로 반전 OP앰프의 NFB(Negative Feedback)에 디지탈입력의 비트수 만큼 저항을 직렬로 구성하고, 스위치의 동작에 의해 부궤환회로의 모습을 변형하여 앰프의 이득을 변화시켜 DAC의 작용을 하는 방식이다. 그림 5에 직렬형 스위치 구성의 DAC를 보였다.

일반적으로, 반전앰프의 경우에 있어서 전압이득은

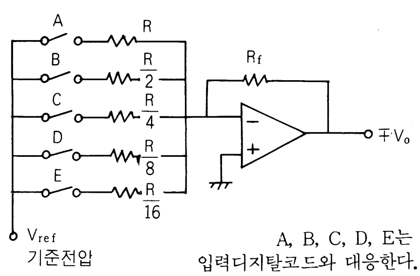
〔그림 2〕 R-2R 래더방식 BCD 코드 DAC



〔그림 3〕 전류가산방식 BCD코드 DAC

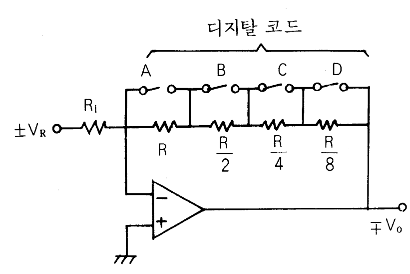


〔그림 4〕 2진 무게치저항을 갖는 DAC



A, B, C, D, E는 입력디지탈코드와 대응한다. A : LSB, E : MSB

〔그림 5〕 직렬스위치 가변이득방식 DAC



$$A_V=-\frac{R_f}{R_i}$$

인데 여기서 R_i는 입력측 저항값이고, R_f는 출력에서 반전입력으로 들어가는 부궤환 저항이다.

여기서는 R_f가 직렬스위치의 구성에 의하여 변화되므로, 만일 스위치가 0일때 ON이 되고 1일때 OFF 되면

$$R_f=R+\frac{R}{2}+\frac{R}{4}+\cdots+\frac{R}{2^n}$$

이 되고 n은 입력 디지탈코드 비트수 -1이 된다. 즉, 4비트이면 n=3이 된다.

그림 6은 병렬형스위치 가변이득방식의 그림이다. 병렬스위치 가변이득 방식은 브리지앰프의 전저항을 $\frac{R}{2^n}$로 모두 같게 하고 반전입력과 접지사이에 중첩형 저항트리를 삽입한 것으로 아날로그 스위치의 한쪽을 접지한다는 점이 특징이다.

이와 같이 OP 앰프를 이용한 직·병렬 가변이득DAC는 입력측 임피던스가 스위칭에 관계없이 항상 일정하다는 것이 특징이다.

4. 정전류원 스위칭방식

일반적으로 정전류원 스위칭방식은 고속의 DAC에 사용되는 DAC이다. 여기에는 등정전류원 방식과 중첩정전류원 방식이 있다.

이와 같은 방법은 주로 모놀리틱 IC로 사용되는 경우가 많다.

그림 7, 그림 8에서 각각 등정전류원 방식과 중첩정전류원 방식의 DAC 원리도를 보였다.

R-2R 래더방식과 가산기방식을 비교해 보기 바란다.

5. 펄스적분방식

이 방식은 입력 디지탈코드를 펄스의 듀티비(Duty Ratio)로 변환하여 기준전압을 ON / OFF하여 그 출력을 저역필터(Low Pass Filter)로 출력하는 형식의 DAC이다. 이것은 비록 변환속도가 느린 단점이 있으나 정밀한 부품을 필요치 않고 일반적인 회로 구성만으로도 만들어지는 잇점이 있다. 그림 9에 이의 원리도를 보였다.

그림 10에 보인 파형을 보면 듀티비를 이용한 평균전압의 가변원리를 쉽게 이해할 수 있다.

듀티비란 출력펄스의 한주기 동안 ON의 시간과 OFF의 시간을 퍼센트로 나타낸 것이다. 따라서 ON의 시간이 증가할수록 평균전압은 증가하고 OFF시간이 늘어나면 당연히 평균전압은 낮아진다. 단, 이때 입력 디지탈코드의 변화가 매우 심하면 출력에 있어서 리플제거율과 응답스피드가 양립될 수 없으므로 그런 경우 고차의 저역필터를 사용해야 한다.

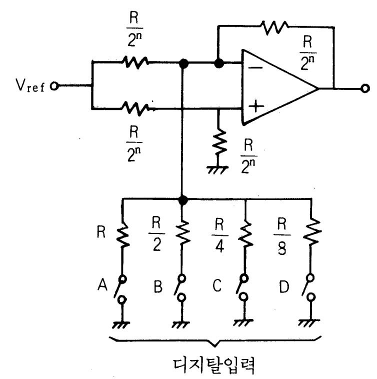
DAC의 출력형식과 입력디지탈코드

앞에서도 R-2R 래더방식의 DAC를 설명할때 간단히 BCD의 경우와 스트레이트 바이너리 코드일 경우를 생각해 보았다. 여기에 입력되는 디지탈코드의 선정은 시스템 설계시 경우에 따라 혹은 해당조건에 따라 달라질 수 있다. 여기서 몇가지 종류를 살펴보면 다음과 같다.

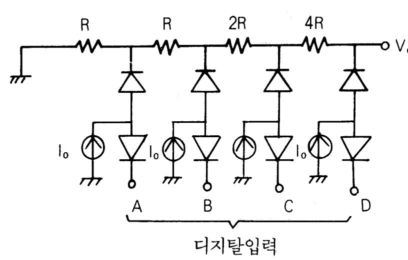
1. 스트레이트 바이너리코드(Straight Binary Code)
2. 스트레이트 바이너리 코드+오프셋(Straight Binary Code+Offset)
3. 오버플로우 비트를 갖는 스트레이트 바이너리 코드(Straight Binary Code With Overflow Bit)
4. 사인비트를 갖는 스트레이트 바이너리 코드(Straight Binary Code With Sine Bit)
5. 오프셋 바이너리 코드(Offset Binary Code)

등이 있는데 이는 DAC의 출력형식과 관계가 있다.

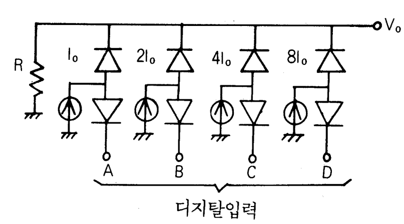
〔그림 6〕 병렬스위치 가변이득방식 DAC



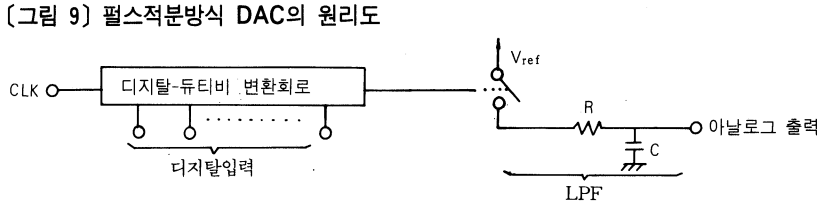
〔그림 7〕 등정전류원방식 DAC



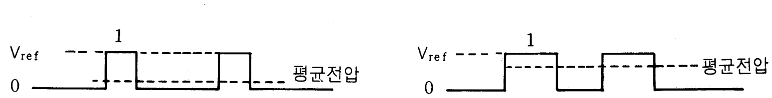
〔그림 8〕 중첩정전류원방식 DAC



〔그림 9〕 펄스적분방식 DAC의 원리도



〔그림 10〕 ON시간이 길어질수록 평균전압은 증가

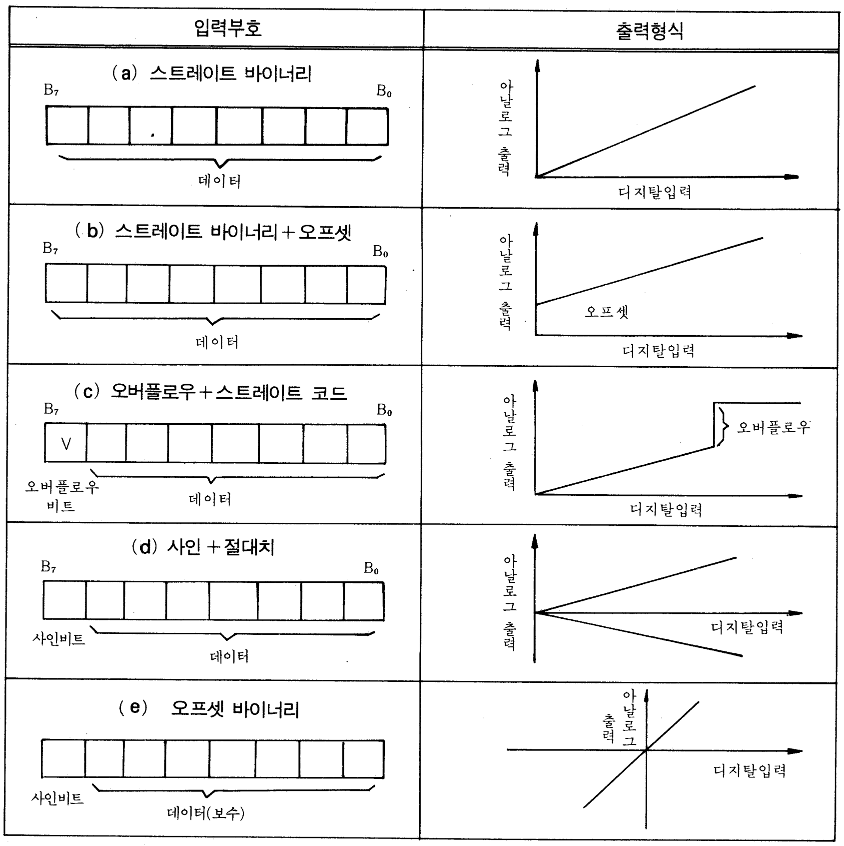


스트레이트 바이너리코드의 경우 입력 디지탈코드에 대한 플러스 방향으로만 선형적 직선관계가 성립하고, 오프셋을 갖는 스트레이트 바이너리 코드의 경우는 오프셋을 갖는다. 즉 일반적인 스트레이트코드에 특정 바이어스를 띄운 상태의 출력을 얻게 된다. 사인비트를 갖는 경우 해당 디지탈입력에 대해 사인비트에 따라서 정·부의 출력을 얻을 수 있고 오프셋 바이너리의 경우는 데이터비트를 보수적 형태로 출력하여 마이너스 방향의 출력을 얻을 수 있다.

이와같이 입력 디지탈코드의 종류는 사용하고자 하는 출력의 종류에 따라 결정되어야 한다.

그림 11은 입력 디지탈코드와 출력의 형식을 나타낸다.

〔그림 11〕 DAC의 입력코드와 출력의 관계



DAC의 선정

앞에서 간단하게 DAC의 종류와 동작원리 등을 설명하였으나 실제로 DAC를 제작하려는 경우 어떠한 사항을 고려하여 DAC를 선정해야 할 것인가가 무척 망설여지는 경우가 많을 것이다. 물론 동작원리를 설명하는 부분에서 선정의 기준을 간단히 나타냈지만 결국 아마츄어 입장에서의 설계에는 자신이 가장 정확히 이해하고 비교적 쉬운 방법으로 제작할 수 있는 방법을 선택하게 될 것이다. 그러나 정식으로 DAC를 설계하는 경우 다음 사항이 고려되야 한다.

① 사용목적
② 안정화 시간
③ 출력범위
④ 분해능력
⑤ 정밀도
⑥ 공급전원
⑦ 동작환경(주위온도, 습도)
⑧ 보존환경(보존온도, 습도)
⑨ 디지탈입력의 특성

위의 열거 사항들을 따로 따로 분리하여 생각할 수는 없지만 무엇보다 신경을 써서 고려해야 할 사항은 당연히 사용목적이 된다. 이 사항의 결정은 결국 고속 DAC를 사용해야 하는가 아니면 저속 DAC를 사용해야 하는가 등을 결정한다.

세틀링타임을 결정한 다음에 전압 출력형인가 혹은 전류 출력형인가를 결정한다. 만일, 세틀링 시간이 짧은 고속의 경우라면 정전류형 DAC를 선정해야 하고 비교적 저속의 경우라면 펄스폭 변조형 DAC를 사용해야 한다.

또, 출력범위의 경우와 모터의 속도 제어의 경우에 역회전 등을 고려한 설계라면, 정·부의 바이폴라 스타일의 출력형식이 결정되어져야 할 것이고 넓은 동작범위에서의 리니어한 동작특성을 구현하기 위해서는 넓은 다이나믹 레인지를 갖는 DAC를 선정해야한다. 즉, 콤플라이언스(Compliance) 전압을 고려해야 할 것이다. 이러한 다이나믹 레인지의 문제를 고려하다 보면 자연히 출력앰프의 전원전압을 고려하게 될 것이고 또한 "출력의 형식이 전류형인가?", "전압형인가?"에 따르는 앰프의 선정도 고려하게될 것이다. 그런데 DAC의 경우는 ADC와는 달리 양자화 오차*1가 원칙적으로 발생하지 않기 때문에 라운딩에러*2나 트렁케이션 에러*3에 의한 비트수를 고려하지 않아도 된다.

그러나 출력의 형식이 고분해 등을 요구하는 경우 정전류인 DAC를 사용한다.

또, 컴퓨터와 인터페이스의 경우 특별히 고려되어야 할 사항은 컴퓨터가 DAC에게 어떠한 코드형식으로 드라이브할 것인가를 정확히 규정해야 한다. 즉 앞에서 설명한 바와 같이 "BCD 코드로 제어가 될 것인가?" 혹은 "스

트레이트 바이너리 코드인가?" 혹은 "사인비트를 갖는 출력형식인가?"를 판단하여 코드에 적절한 DAC형식을 갖추어야 한다. 그외의 선정요령이라면 "현실적으로 입수가 용이한가?", "가격이 저렴한가?"등을 고려해야 할 것이다.

일반적으로 DAC의 정밀도를 결정하는 중요한 열쇠는 그 방법의 선택보다는 최종적으로 합성되어 출력으로 나오는 앰프의 설계를 간과할 수 없다. 대부분 출력앰프는 OP앰프를 사용하는 경우가 많은데 이 경우 파라미터 결정의 핵심이 되는 R과 C의 선정이 핵심이된다.

즉, 저항의 경우 가능한 1%내의 금속피막저항을 사용하는 것이 좋고 캐패시터의 경우 누설전류가 적은 것을 사용하는 것이 좋다.

이외에도 기준전압의 내장여부나 온도 드리프트에 관한 특성, 미분 비직선성 등 여러가지 고려의 대상이 있다. 이러한 모든 것을 한꺼번에 고려하는 것은 여간 성가신 일이 아니므로 가능한 가장 중요하게 판단되는 몇가지 요소를 선정하여 그 선정된 사양을 만족하는 DAC의 설계가 바람직하다.

그러면 지금까지의 내용을 토대로 간단한 DAC를 제작해 보기로 하겠다.

여기서 우선 DAC를 제작하여 컴퓨터로 발생되는 코드의 변화에 대한 출력의 변화만을 고려하므로 앞에서 설명한 까다로운 조건들은 고려의 대상에서 제외한다.

DAC의 원리적동작을 가장 간단하고 확실하게 살필 수 있는 방법은 앞에서 언급된 R-2R 래더방식이다.

이것은 큰 정밀도를 요구하지 않는 범위에서 무난히 제작될 수 있는 방법이다.

응용

DAC를 컴퓨터와 인터페이스하는 방법은 본지 88년 12월호를 참조하기 바란다(그림 12, 13 참조).

우선 MSX와의 인터페이스를 살펴보면, 8255AP-5의 A, B, C중 어느 포트에 관계없이 모든 포트나 특정 포트를 출력으로 세트하고 그 포트에 그림 12를 접속하기만 하면 된다.

발생되는 출력의 모습은 오실로스코프를 통하여 보는 것이 가장 좋으나 그것이 없는 경우 출력단에 테스터를 대고 코드의 변화를 천천히 하면 출력되는 전압을 볼 수 있다.

IBM-PC의 경우도 LS374와 그림 12를 그림 13에 보인 형태로 접속하면 되고 출력의 형태도 위에서 설명한 동일한 방법을 사용한다. 이렇게 발생되는 출력은 파형발생기나 모터의 속도 제어 등에 사용되어 질 수 있다. 리스트 1에 이의 예제 프로그램을 보였다.

맺음말

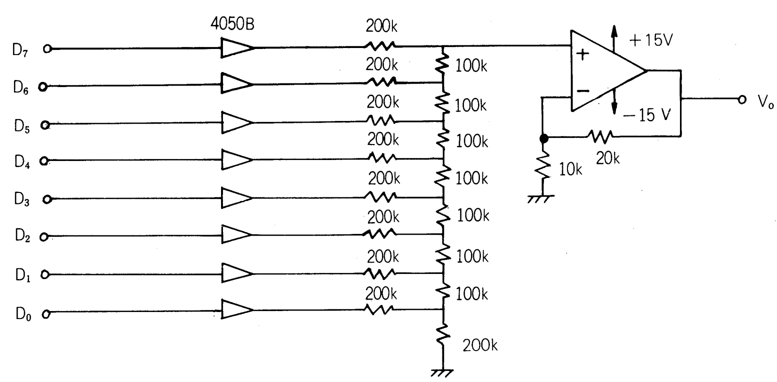
이제까지 간단하나마 ADC와 DAC에 관하여 설명하였다. 지금까지의 간단한 설명으로 독자가 원하는 퍼스널 계측시스템을 제작할 수 있으리라고 생각한다.

실제적으로 정밀한 계측시스템을 제작하려면 ADC와 DAC외의 많은 부분을 고려해야 한다.

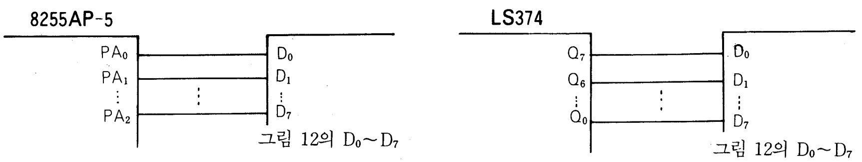
노이즈제거를 위한 필터링의 방법이라든가, 미소신호의 발생이나 계측에 필요한 프리앰프의 선정 등 다양한 지식들이 요구된다. 이러한 내용들은 많은 경험을 필요로 하는 부분들 일 것이다.

아뭏든 지금까지의 글이 ADC나 DAC

〔그림 12〕 R-2R 래더방식을 사용한 DAC



〔그림 13〕



(a) MSX의 경우 (b) IBM-PC의 경우

*1 양자화 오차 (Quantization Error): 양자화에 의하여 발생하는 오차. 양자화의 구간 폭을 좁게 하면 오차는 줄어 들지만 필요한 비트 수가 늘어난다.
*2 라운딩 오차(Rounding Error): 어떤 수치를 라운딩함으로써 참값에 대해 발생하는 오차
*3 절단 에러(Truncation Error): 무한 급수를 유한급수로 바꾸거나, 무한히 계산해야 할 수치 처리의 반복 과정을 어느 한도에서 정지할 경우에 나타나는 에러.

를 이해하려 했던 분들께 작은 도움이나마 됐으면 하는 바램이며, 장황한 글을 끝까지 참을성있게 읽어 준 분들께 감사를 드린다.

〈참고문헌〉

1. A-D / D-A 변환회로의 설계와 응용, 김태영 역, 우정출판사
2. 마이크로 컴퓨터 응용계측회로, 이문기 저, 기술연구사
3. 리니어 IC의 응용과 실제, 한영신 · 김경호 편저, 전자기술사
4. Introduction to Digital Control Systems, Hugh F. Vanlandingham, Macmillian Publishing Company
5. 입문에서 응용까지 OP 앰프 실전기술, 谷本茂 저, 세운출판사

□ 알림 □

지금까지 인터페이스 강좌를 읽어주신 독자여러분께 감사드립니다. 인터페이스 강좌는 이번회를 마지막으로 마치게 되었으며 다음호부터는 새로운 연재강좌인 '하드웨어 강좌'를 게재하고자 합니다. 새로 연재될 하드웨어 강좌는 하드웨어에 전혀 지식이 없는 사람도 쉽게 접근할 수 있도록 재미있게 구성하고자 하오니 앞으로도 많은 애독을 바랍니다.

〔리스트 1〕

```
100 ' R-2R LADDER RESISTOR TYPE DAC TEST PROGRAM
110 ' Programed by C.S.W
120 ' 88/12/26
130 ' IBM-PC XT Version
140 '
150 PO=&H580
170 FOR X=0 TO 255
180 PRINT RIGHT$("00"+HEX$(X),2);" ";
190 OUT PO,X
200 X$=INPUT$(1)                        :'For Step operation
210 NEXT X
220 '
230 PI=ATN(1)*4                         :'Define Pi
240 FOR X=0 TO 2*PI STEP .01            :'Loop for one cycle
250 Y=INT(SIN(X)*100)+128               :'Sine Generation
260 PRINT RIGHT$("00"+HEX$(Y),2);" ";   :'Display sine code
270 OUT PO,Y                            :'SINE WAVE GENERATION
280 NEXT X                              :'Loop for one cycle
```

(a) IBM

```
100 ' R-2R LADDER RESISTOR TYPE DAC TEST PROGRAM
110 ' Programed by C.S.W
120 ' 88/12/26
130 ' MSX Version
140 '
150 CR=&HDB:PA=&HD8:PB=&HD9:PC=&HDA
160 OUT CR,&H80                         :'Set all ports to Output
170 FOR X=0 TO 255
180 PRINT RIGHT$("00"+HEX$(X),2);" ";
190 OUT PA,X
200 X$=INPUT$(1)                        :'For Step operation
210 NEXT X
220 '
230 PI=ATN(1)*4                         :'Define Pi
240 FOR X=0 TO 2*PI STEP .01            :'Loop for one cycle
250 Y=INT(SIN(X)*100)+128               :'Sine Generation
260 PRINT RIGHT$("00"+HEX$(Y),2);" ";   :'Display sine code
270 OUT PA,Y                            :'SINE WAVE GENERATION
280 NEXT X                              :'Loop for one cycle
290 GOTO 240                            :'Repeat Generation
```

(b) MSX

그래픽

이기성/장왕교재연구원 상무

컴퓨터 그래픽이란 말이 널리 퍼지고는 있지만 아직도 컴퓨터 내부에서 어떻게 그래픽을 생성시키며 스스로 어떻게 그래픽을 구현해 볼 것인가에 대해 궁금한 독자들이 많을 것이다. IBM PC의 BASIC언어는 많은 그래픽기능들을 갖고 있어서 그래픽에 대한 독자들의 욕구를 쉽게 충족시키기에 최적의 언어라고 생각된다.

이번 연재에서는 그래픽에 대해서 공부해 보기로 하자.

화면모드

모니터 화면에 디스플레이되는 모드(Mode : 상태)는 크게 두가지가 있다. 문자들만 주로 사용하는 텍스트 모드(Text Mode)와 그림을 그리는데 주로 사용하는 그래픽모드(Graphic Mode)가 그것이다.

텍스트모드에서의 그래픽은 그래픽모드에 비해 이미지를 만들어내는 속도가 빠르다는 점과 칼라를 지정할 수 있는 기능, 그리고 섬세한 애니메이션 기능 등을 갖고 있기는 하지만 개개의 화면 픽셀을 콘트롤하는 기능과 선분, 각종 호등을 그릴 수 있는 기능은 갖추고 있지 않다. 텍스트모드의 화면은 다시 가로로 80칸, 세로로 25줄(WIDTH 80)인 경우와 가로로 40칸, 세로로 25줄(WIDTH 40)인 경우가 있다.

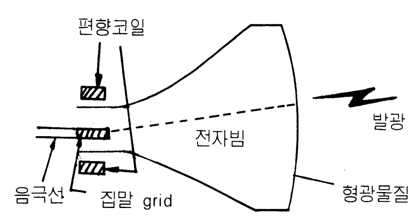
한편 그래픽모드에서는 화면을 구성하는 방식에 따라 중해상도 그래픽모드(SCREEN 1)와 고해상도 그래픽모드(SCREEN 2)가 있다. IBM PC에서는 래스터 스캔방식(Raster Scan Method) 으로 그래픽을 만들어 낸다. 이는 TV의 주사방법과 같이 음극선에서 발생시킨 전자빔을 편향코일을 사용하여 모니터화면에 좌에서 우로, 위에서부터 아래로 주사해 가는 방식으로 한번에 한줄씩 디스플레이 한다(그림 1 참조). 즉 디스플레이되는 화면은 아주 조밀하게 짜여진 바둑판과 같이 구성되며 가로 640, 세로 200인 화면을 고해상도 그래픽모드, 가로 320, 세로 200인 화면을 중해상도 그래픽모드라 한다. 참고로, 이들 화면을 구성하는 최소 단위인 하나의 점을 픽셀(PIXEL : Picture Element, 화소)이라고 부르고 있다.

컴퓨터 그래픽은 이러한 각 픽셀들을 제어함으로써 가능해지는데 BASIC 언어는 PSET과 PRESET이란 명령으로 이같은 기능을 수행하고 있다. PSET (X,Y)란 Y번째 줄(래스터 스캔라인)의 X번째 픽셀을 ON시키는 것과 같다. 예를 들어, 고해상도 그래픽모드에서 PSET(0,0)는 화면의 맨 좌측상단의 픽셀하나를 ON시키고 PSET (639, 199)는 화면의 맨 우측하단의 픽셀을 ON시킨다. 그리고 PRESET (X,Y)는 지정 위치의 픽셀을 배경색으로 찍어주므로 점을 지우는 역할을 한다.

이번에는 위에서 설명한 화면모드를 선택하는 명령과 각각의 화면모드에서의 그래픽에 대해 알아 보자.

화면모드를 선택하는 명령어는 SCREEN 이다. 이 명령과 함께 모드번호, 칼라지정, 해당 페이지번호등의 파라미터가 선택적으로 쓰일 수 있으며 그 형식을 보면 다음과 같다.

〔그림 1〕 래스터방식의 디스플레이




```
SCREEN [mode][,[burst][,[apage][,vpage]]]
```

mode = 0(텍스트모드)
1(중해상도 그래픽모드)
2(고해상도 그래픽모드)

burst : 텍스트모드에서 0이면 칼라 ON, 0를 제외한 숫자가 오면 칼라 OFF(흑백 지정)를 지정하는데 그래픽모드에서는 반대로 된다.

apage : 'active screen page'로 다음 순서에 디스플레이 될 페이지들 지정한다.

vpage : 'visual screen page'로 현재 디스플레이되는 페이지를 나타낸다.

여기서 모든 선택사항들을 항상 지정할 필요는 없으며 필요한 부분만 선택하면 된다. 보통은 텍스트모드인 경우는 SCREEN 0, 그래픽모드인 경우는 SCREEN 1(또는 2)으로 지정하여 사용해도 된다. 그리고 화면 페이지는 메모리-맵에서 설명되므로 여기서는 무시해도 좋다.

텍스트모드(그래픽 문자)

텍스트모드에서의 그래픽은 어느정도 제한되어 있다. 왜냐하면 그래픽모

드에서와 같이 픽셀단위로 그래픽을 만드는 것이 아니라 ROM속에 이미 기억되어 있는 블록단위의 선 또는 무늬만을 이용해야 되기 때문이다. 그러나 그래픽모드에서는 제공하기 힘든 빠른 그래픽 처리속도, 다양한 색상 제공 및 부드러운 에니메이션이 가능하다는 장점도 갖고 있다.

ROM에는 각종 기호 폰트를 저장하고 있는 256가지의 캐릭터 세트가 있다. 여기서 하위 128문자는 ASCII문자로 이미 잘 알려진 키보드내의 문자들이며 상위 128문자는 각종 선, 무늬 등을 나타내는 그래픽문자이기 때문에 텍스트모드에서의 그래픽은 바로 이 상위 128문자세트를 이용하는 것으로 볼 수 있다.

그럼, ROM속에 있는 문자들을 어떻게 화면으로 나타낼 수 있을까? 베이직은 이를 위해 CHR$라는 명령어를 준비하고 있다. CHR$는 원하는 번호(ASCII 코드)에 해당하는 문자를 화면에 표시해 주는 기능을 갖는다.

```
CHR$(n)
```

n=문자번호(0~255)

예를 들어 화면에 A란 문자를 나타내기 위해 베이직 프롬프트(_)상에서 PRINT CHR$(65)를 입력한 후 〈ENTER〉 키를 누르면 원하는 문자(A)가 즉시 출력된다. 물론 이런 경우는 CHR$ 명령없이 키보드의 해당 키를 누르면 된다. 하지만 ║ 또는 ╦ 과 같은 그래픽문자들은 키보드에 나타나 있지 않기 때문에 CHR$ 명령을 이용해야 함을 알아 두자. **리스트** 1은 이 CHR$ 명령을 이용하여 화면에 그래픽문자들을 출력시키는 프로그램이다. 이 글을 읽는 독자들도 자기가 보유하고 있는 컴퓨터에 어떤 그래픽 문자들이 저장되어 있는지 리스트 1을 입력하여 테스트 해보기 바란다. **화면** 1은 필자가 가지고 있는 컴퓨터에서 출력한 예이다.

〔리스트 1〕 ASCII.BAS

```
1 'ASCII.BAS program
5 '-------------------------------------------
10 SCREEN 0,1: KEY OFF: CLS
20 FOR I = 169 TO 223 STEP 5
30     FOR J = 0 TO 4
40       PRINT "    ";I + J;":";
50       PRINT " ";CHR$(I + J);
60     NEXT J
70     PRINT : PRINT
80 NEXT I
90 '-------------------------------------------
```

〔화면 1〕 각종 그래픽 문자

```
169 : ⌐    170 : ¬    171 : ½    172 : ¼    173 : ¡
174 : «    175 : »    176 : ░    177 : ▒    178 : ▓
179 : │    180 : ┤    181 : ╡    182 : ╢    183 : ╖
184 : ╕    185 : ╣    186 : ║    187 : ╗    188 : ╝
189 : ╜    190 : ╛    191 : ┐    192 : └    193 : ┴
194 : ┬    195 : ├    196 : ─    197 : ┼    198 : ╞
199 : ╟    200 : ╚    201 : ╔    202 : ╩    203 : ╦
204 : ╠    205 : ═    206 : ╬    207 : ╧    208 : ╨
209 : ╤    210 : ╥    211 : ╙    212 : ╘    213 : ╒
214 : ╓    215 : ╫    216 : ╪    217 : ┘    218 : ┌
219 : █    220 : ▄    221 : ▌    222 : ▐    223 : ▀
```

어느정도 프로그래밍 경험이 있는 독자는 화면에 복잡한 이미지를 표현하기 위해서 일일이 CHR$ 명령을 이용한다는 것은 비효율적임을 알 것이다. 그래서 실제의 텍스트모드 그래픽에는 〈Alt〉키를 이용하는 경우가 대부분이다. 즉 프로그램 코딩에 직접 원하는 형상을 그리는 방법으로 다음과 같이 〈Alt〉키와 그래픽문자의 코드번호를 동시에 입력하는 것이다.

1. 〈Alt〉키를 누른다.
2. 〈Alt〉키를 누른 상태에서 키보드의 오른쪽에 있는 계산용 키-패드의 숫자키를 이용하여 218을 입력한다.
3. 〈Alt〉키를 놓는다.

계속해서 이번엔 〈Alt〉키를 누른 상태에서 196을 입력한 후 〈Alt〉키를 놓는다. 화면을 보면 각 숫자에 해당하는 그래픽 문자가 출력되는 것을 볼 수 있다. 리스트 2는 이런 방법으로 재미있는 그래픽을 만들어 본 예이다.

독자들 스스로가 컴퓨터 게임용 프로그램에 볼 수 있는 여러가지 형상들과 막대 그래프 등을 만들어 보기 바란다.

그래픽모드(픽셀)

앞에서도 언급했지만 이 모드에서는 픽셀단위로 그래픽이 처리되기 때문에 섬세한 이미지를 마음대로 만들 수 있다.

SCREEN 1과 SCREEN 2의 차이점은 고해상도 그래픽모드에서는 메모리의 1비트가 화면의 픽셀 하나와 연결되므로 칼라를 지정할 수 없으나 중해상도 그래픽모드에서는 고해상도의 픽

〔리스트 2〕 〈Alt〉키를 이용한 그래픽

```
1 'ALT-USE.BAS program
5 '----------------------------------------
10 SCREEN 0: WIDTH 80: KEY OFF: CLS
20 LOCATE 5,20: PRINT "                        "
30 LOCATE 6,20: PRINT "    ┌──────────────┐    "
40 LOCATE 7,20: PRINT "    │ A B C      █ "
50 LOCATE 8,20: PRINT "    └──────────────████  "
60 LOCATE 9,20: PRINT "  ┘│  ■  │  ■  │└      "
70 LOCATE 10,20:PRINT "   └  ┌─────┐  ┘       "
80 LOCATE 11,20:PRINT "     └──────────┘       "
90 LOCATE 12,20:PRINT " ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ "
100 LOCATE 13,20:PRINT " ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ "
300 PLAY "C D E"
```

〔표 1〕 배경색 및 팔레트 색상

코드	색 상
0	흑색(Black)
1	청색(Blue)
2	녹색(Green)
3	청색(Cyan)
4	적색(Red)
5	자주색(Magenta)
6	갈색(Brown)
7	백색(White)
8	회색(Gray)
9	밝은 청색(Light Blue)
10	밝은 녹색(Light Green)
11	밝은 청색(Light Cyan)
12	밝은 적색(Light Red)
13	밝은 자주색(Light Magenta)
14	노랑색(Yellow)
15	밝은 백색(High-intensity White)

셀 2개가 모여 하나의 픽셀로 처리되므로 이 두개의 비트조합으로 제한적이나마 칼라지정이 가능하다는 점이다. (대신 고해상도에서는 가로로 640개의 픽셀을 이용하여 보다 섬세한 그래픽이 가능해진다.)

중해상도 그래픽모드에서 COLOR 명령어는 다음과 같은 형식으로 주어진다.

COLOR [배경색][,팔레트]

배경색 :16개의 값중에서 하나(0~15)를 선택할 수 있다.
팔레트 : 0과 1중 하나를 선택한다.

하나의 팔레트 값을 선택했을 때 그 전경색으로는 3가지를 선택할 수 있으므로 화면에 동시에 나타낼 수 있는 색상은 배경색까지 합하여 4가지 색상이 된다(표 1 참조).

그러나 이글을 읽고 있는 독자가 모노크롬 모니터를 갖고 있다면 전경색으로 1 또는 3을 지정하는 것이 무리가 없을 것이다.

(1) 화면에 점찍기

가장 기본적인 그래픽 명령으로 PSET과 PRESET이 있다. 이들 명령어는 개개의 픽셀들을 주어진 팔레트로부터 색상을 ON 또는 OFF시킴으로써 화면상에 점을 찍고 지우는 역할을 하고 있다.

PSET(X,Y) [,색상]
PRESET(X,Y) [,색상]

(x,y): 점을 찍거나 지울 위치를 지정하며 화면의 맨 좌측상단이 (0,0)이 된다.
색상 :주어진 팔레트로부터 선택할 색상코드 번호이다. SCREEN 2에서는 색상을 지정할 수 없으므로 검은색(Black)은 0과 2를, 흰색(White)은 1과 3을 선택한다.

여기에서 알 수 있듯이 PSET과 PRESET은 똑같은 기능(픽셀을 지정한 색상으로 ON시키는 것)을 수행하고 있으며 다만, PSET은 디폴트로 칼라코드 3(SCREEN 2에서)을, PRESET은 디폴트로 칼라코드 0을 취하고 있을 뿐이다.

리스트 3는 PSET명령만을 가지고 이미지를 만들고 있다. 즉 40번 문장에서 계산한 줄의 임의의 위치에 픽셀(점 : 우주를 상징)들이 찍히게 하였다.

(2) 선긋기(LINE, DRAW)

임의의 선분을 그릴때는 LINE명령을 이용하며 뒤에 따라오는 파라미터들을 잘 응용하면 여러가지 효과를 얻을 수 있다.

팔레트 0		팔레트 1	
0	배경색	0	배경색
1	녹 색	1	청 색
2	적 색	2	자주색
3	갈 색	3	백 색

LINE [X1,Y1] - (X2,Y2) [,[색상][, B[F]]
[, 스타일]]

[X1, Y1]:직선의 시작점으로 이미 지정되어 있다면 생략해도 된다.
(X2, Y2):직선의 끝나는 점
B[F]:"B"(Box)를 지정하면 앞에서 지정된 두 위치를 대각선으로 하는 사각형이 그려진다. "F"(Fill)를 지정하면 사각형이 COLOR 문에서 지정된 색으로 칠해진다.
스타일 :지정된 이진배열로 선분이 그려진다.

SCREEN 2모드에서 다음 네 문장을 입력하여 차이점을 살펴보기 바란다.

```
(a) LINE(150,50) - (400,150)
(b) LINE(150,50) - (400,150),,B
(c) LINE(150,50) - (400,150),,BF
(d) LINE(150,50) - (400,150),,B,&HF249
```

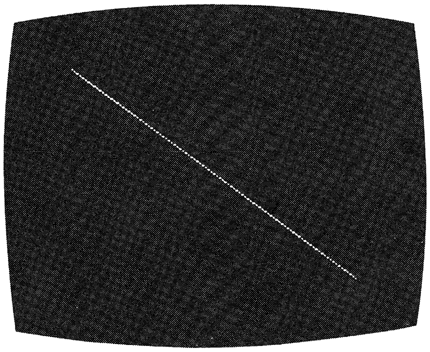
화면 2에 각각의 출력화면을 나타내었다.

화면 2의(d)와 같이 LINE문장과 함께 스타일이 지정되면 여러가지 패턴의 선을 얻을 수 있게 된다. 그 원

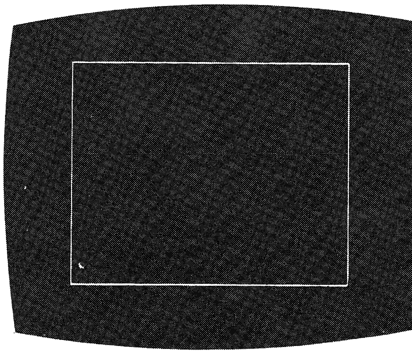
〔리스트 3〕 PSET 명령을 이용한 SPACE.BAS

```
1 'SPACE.BAS program
5 '-----------------------------------------
10 RANDOMIZE TIMER
20 SCREEN 2: KEY OFF: CLS
30 FOR I = 1 TO 100
40     X = RND*640: Y = RND*200
50     PSET (X,Y)
60 NEXT I
70 '-----------------------------------------
```

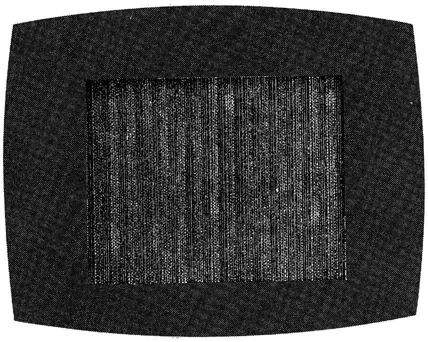
〔화면 2〕 LINE 명령의 각종 출력



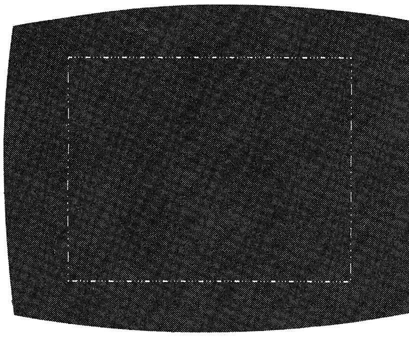
(a) LINE (150, 50) - (400, 150)



(b) LINE (150, 50) - (400, 150),, B



(c) LINE (150, 50) - (400, 150),, BF



(d) LINE (150, 50) - (400, 150),, B, & HF249

리를 보면, 지정한 16진수값이 한 숫자당 4비트의 2진 값으로 대응되어 총 16비트의 배열을 이루게 되며 각 비트들의 ON, OFF에 따라 독특한 패턴이 형성되고 이 배열패턴으로 LINE명령을 수행하고 있는 것이다.

그림 2에 스타일을 형성하는 원리를 나타내었다. **리스트 4**는 LINE명령을 이용한 단순한 프로그램으로 생동감 있는 선의 율동을 느낄 수 있을 것이다.

LINE명령과 원리는 비슷하나 화면 안에서 픽셀단위로 마음대로 돌아다니며 선을 그려주는 DRAW명령이 있다. 그래픽의 마우스라고 불릴 정도로 독특한 성격을 지니는 이 명령은 특히 내부에 또다른 그래픽 스트링이 쓰인다는 점에서 그 기능은 더욱 놀랄만하다.

```
DRAW string$
```

string$: 화면안에서 커서의 움직임을 지시하는 그래픽기호들이 모인 스트링이다.

이 DRAW명령을 익힌다는 말은 곧 그래픽 기호들로 구성된 스트링을 익힌다는 말이 된다. 가장 흔히 쓰이는 기호로 U(Up), D(Down), R(Right), L(Left) 등이 있다. 즉

```
DRAW "U10 R10 D10 L10"
```

은 가로 세로 픽셀이 각각 10인 정사각형을 그린다. (실제로 실행시킨 결과 직사각형처럼 보이는 것은 가로 세로의 픽셀 수가 틀린 화면의 속성 때문이다.) 위 문장을 말로 표현해 보면 "기준점에서 위로 길이 10(실제는 픽셀 수)만큼 선을 긋고(U10), 오른쪽으로 길이 10만큼 긋고(R10) 다시 밑으로 10만큼 긋고(D10) 왼쪽으로 10만큼 선을 그어라(L10)"가 된다. 그럼 여기서 그림을 그려나가는 시작점, 즉 기준점은 어떻게 나타낼까? 물론 PSET (X,Y)로 정해도 되지만 여기서는 M이란 기호가 준비되어 있다. Mx,y라고 지정하면 (x,y) 지점부터 시작한다. 그리고 이같은 절대좌표가 아닌 상대좌표 지정도 가능하며 이 경우는 M+x, +y로 나타낸다. 이 외에도 원하는 위치까지 선을 긋지 않고 위치지정만 나타내는 B(Blank), 지정한 선을 긋고 다시 원래의 커서 위치로 돌아오는 N, 각도를 나타내는 A, 스케일 변환을 나타내는 S기호등이 있다(**표 2** 참조).

리스트 5는 독자들이 DRAW명령을 이해하는데 좋은 예가 될 것으로 믿는다. 여기서 보면 40~80번 문장에서 몸체를 그려주고 있고 화면 중앙을 지정하기 위해서 BM160, 100을 지정하고 있다. (그러나 실제로는 BM−4, +2에 의해(156, 102)지점에서 그림을 그리기 시작한다.) 또한 FOR~NEXT문에 의해 제어되는 외부 변수(ANGLE, S)를 DRAW문장에 도입, 스케일변환은 물론 90° 간격으로 변화되도록 했다.

〔그림 2〕 LINE 스타일

LINE 패턴																
2진수 패턴	1	1	1	1	0	0	1	0	0	1	0	0	1	0	0	1
16진수	F				2				4				9			

〔리스트 4〕 LINE 명령을 이용한 LINESHOW.BAS

```
1 'LINESHOW.BAS program
5 '-----------------------------------
10 SCREEN 1,0: KEY OFF: CLS
20 RANDOMIZE TIMER
30 X1 = 320*RND: Y1 = 200*RND
40 X2 = 320*RND: Y2 = 200*RND
50 LINE(X1,Y1) - (X2,Y2)
60 GOTO 30:'---------------------------
```

〔표 2〕 DRAW 명령에 쓰이는 각종 그래픽 기호

제어명령	형 식	동 작	파라미터
이동 및 선긋기	M x, y	점(x, y)를 지정한다.	x, y=절대 좌표치
	M +x, +y	점(X+x, Y+y)를 지정한다.	X, Y=가장 최근에 지정된 점 +x, +y=상대 좌표치
	Un	Up(↑)	n=줄 수
	Dn	Down(↓)	n=줄 수
	Rn	Right(→)	n=칼럼 수
	Ln	Left(←)	n=칼럼 수
	En	대각선방향(↗)	n=대각선 거리
	Fn	대각선방향(↘)	n=대각선 거리
	Gn	대각선방향(↙)	n=대각선 거리
	Hn	대각선방향(↖)	n=대각선 거리
표 제어	B	다음 지점까지 선을 긋지 않고 이동.	
	N	한 동작을 행한 다음 원래 위치로 복귀.	
동작	An	각도를 변경한다.	n=회전각도 0=0° 1=90° 2=180° 3=270°
	TAn	각도 변경	n=회전각 −360≦n≦360° n>0 반시계방향 n<0 시계방향
	Sn	스케일 변환	1≦n≦255
	Cn	주어진 팔레트에서 칼라 선택	n=0 또는 1(640×200 모드) =0, 1, 2 또는 3(320×200 모드)
	Point, bound	특정영역을 칠하기	
	X 변수	외부의 명령 스트링을 실행	

(3) 원 그리기(CIRCLE)

그래픽 이미지를 만들때 종종 원, 타원, 호등의 필요성을 느낀다. 이 모든 것들이 CIRCLE명령 하나로 가능하다면 독자들은 믿겠는가?

CIRCLE(X,Y),반지름 [,색상[,시작, 끝[,비율]]]

(X,Y): 원, 타원 또는 호의 중심 위치
R: 반경
시작, 끝 : 원주 또는 호의 시작과 끝을 나타내는 라디안 각
비율 : 타원을 그릴때의 가로 세로 비

일반적으로 정상원을 그릴때는 반경만을 지정한다. 그림 3에 시작점, 끝점 및 라디안 각을 그림으로 보였다. 이러한 파라미터들을 잘 활용하면 여러가지 효과를 얻을 수 있으므로 다음 문장들을 차례로 실행시켜 보기 바란다.

```
(a) CIRCLE(300,100),80
(b) CIRCLE(300,100),80,,,,.8
(c) CIRCLE(300,100),80,,,,10/3
(d) CIRCLE(300,100),80,3,-.6,-6
```

리스트 6은 반경을 랜덤하게 처리하는 원 무늬를 그린다(화면 3 참고).

(4) 색칠하기(PAINT)

LINE명령의 BF와 같이 지정영역을 원하는 색상으로 채워주는 명령이 PAINT이다.

PAINT(X,Y) [채우는 색상[, 경계선 색상]]

(X,Y): 지정 색상으로 채워지기 시작하는 위치
채우는 색상 : 색칠하고자 하는 색상이다.
경계선 색상 : 이미지의 경계선을 나타내는 색상이다.

PAINT명령을 이해하기 위해서 앞에서 소개한 리스트 6의 BUBBLE.BAS에 다음 문장을 삽입하여 실행시켜 보자.

```
70 PAINT (X,Y),1
```

애니메이션

이번에는 그래픽이미지를 원하는 위치로 움직이게 하는 애니메이션에 대해서 알아보자. 베이직에서는 애니메이션을 실현하기 위해서 GET, PUT 및 텍스트모드에서의 페이지를 이용할 수 있다. 그래픽모드에서의 애니메이션은 특정한 화면의 이미지를 배열에 비트 단위로 기억했다가(GET) 위치

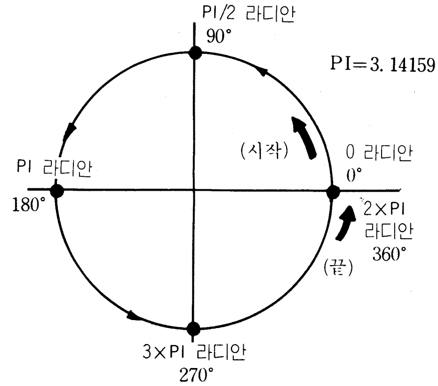
〔리스트 5〕 DRAW 명령을 이용한 GLIDER.BAS

```
1 'GLIDER.BAS program
5 '-----------------------------------------------
10 SCREEN 1,0: CLS: KEY OFF
15 FOR ANGLE = 0 TO 3
20    FOR SCALE = 1 TO 8 STEP .3
30    S = SCALE ^ 2:CLS
40    DRAW "BM160,100 A=ANGLE;S=S; BM-4,+2"
50    DRAW "U4 M+2,-2 R4 M+2,+2 D4 M-2,+2 L4 M-2,-2"
60    DRAW "BU1 M-17,-1 M+17,-1 BR8 M+17,+1 M-17,+1"
70    DRAW "BU3 L8 BU1 BR1 R6 BL3 D1"
80    DRAW "BU2 BL1 M+1,-5 M+1,+5"
90    FOR TIME = 1 TO 300: NEXT TIME
100 NEXT SCALE,ANGLE:'-------------------------------
```

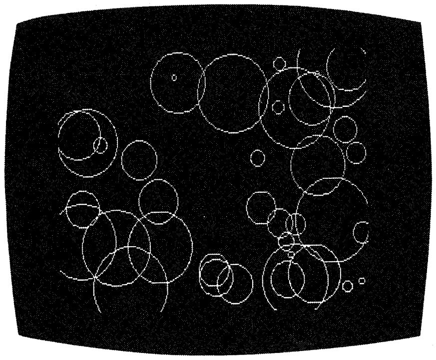
〔리스트 6〕 CIRCLE 명령을 이용한 BUBLE.BAS

```
1 'BUBBLE.BAS program
5 '-------------------------------------
10 SCREEN 1: DEFINT A-Z
20 FOR COUNT = 1 TO 40
30 X = RND*310+5
40 Y = RND*180+5
50 RADIUS = RND*40
60 CIRCLE (X,Y),RADIUS
90 NEXT COUNT
100 END:'-------------------------------
```

〔그림 3〕 CIRCLE 명령에 쓰이는 각도



〔화면 3〕 BUBBLE.BAS 출력



를 이동시키며 다시 그려나가는(PUT) 원리로 구현하고 있다.

```
GET (X1,Y1) - (X2,Y2), 배열명
```

(X1,Y1)–(X2,Y2): 화면의 특정 이미지를 이동시키기 위해 사각형으로 정의한다. (X1,Y1)은 왼쪽 상단, (X2,Y2)는 오른쪽 하단 꼭지점을 나타낸다.
배열명 : 앞에서 정의한 사각형내의 픽셀값들을 기억하기 위한 수치배열명이다.

여기서 배열명은 프로그램 서두에서 DIM문으로 정의되어야 한다. 이때 데이터를 저장하기 위한 배열요소, 즉 이미지를 기억하기 위해 필요한 바이트수를 계산하려면 다음 식을 이용하면 된다.

```
4 + INT((X * 픽셀당 비트수 + 7)/8) * Y
```

X: 수평방향에 있는 픽셀 수
Y: 수직방향에 있는 픽셀 수
픽셀당 비트수 : 중해상도 모드에서는 2, 고해상도 모드에서는 1

```
PUT (X,Y), 배열명 [, 작용]
```

(X,Y): 그려질 이미지의 맨 좌측 상단좌표
배열명 : 특정 이미지가 GET문에 의해 기억되어 있는 수치배열명
작용 : 화면에 이미지를 그리는 작용을 말하며 PSET, PRESET, XOR, OR, AND 등이 있다.

PUT이 들어가는 문장에 작용들을 번갈아가며 실행시켜 보자. 리스트 7은 그래픽 모드에서의 애니메이션을 보이기 위해 작성한 것이다. M1과 M2로 핀과 볼의 이미지를 픽셀단위로 기억시켰으며 60~90번 문장에서 동일한 DRAW명령을 사용하여 핀을 그려주었다. 145번 문장에서부터 움직이는 위치를 제어하는데 이 부분의 FOR~NEXT 루프를 변경시킴으로써 다양한 움직임을 만들 수 있게 된다.

한편 텍스트모드에서의 애니메이션은 프로그래밍 논리에 의한 방법도 있지만 자연스러운 움직임을 얻기 위해 메모리맵의 페이지를 이용하는 것이 좋다.

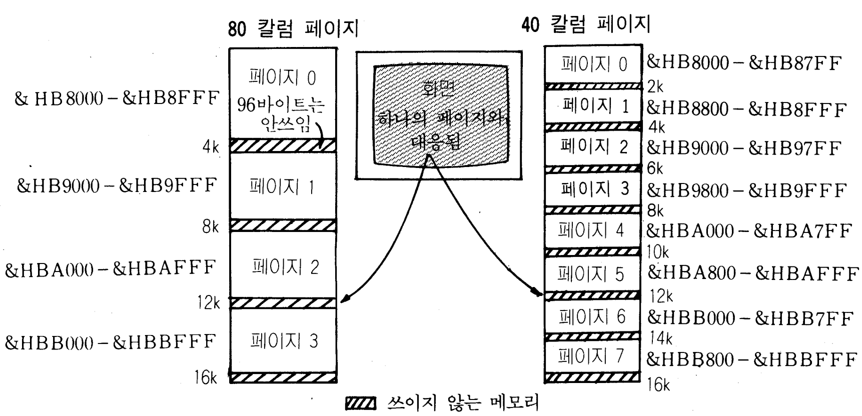
메모리맵과 페이지

칼라 / 그래픽스 어댑터를 사용할 때 텍스트와 그래픽스 화면은 메모리 위치 &HB8000에서 &HBBFFF까지의 16K바이트에서 디스플레이 된다. 하지만 이 16K바이트 용량이 한번에

〔리스트 7〕 그래픽 모드에서의 애니메이션

```
1 'BALLING program
5 '----------------------------------------------------------
10 DIM M1(300),M2(300)
20 SCREEN 2: KEY OFF: CLS
30 'Draw the floor
40 LINE (0,110) - (640,113),,BF
50 LINE (0,116) - (639,120),,BF
60 FOR COUNT = 0 TO 3
70     READ X1
80     PSET(X1,109): GOSUB 250
90 NEXT COUNT
100 'Store the picture
110 GET(490,75)-(610,109),M1
120 DATA 500,530,560,590
130 CIRCLE(100,100),25
140 GET(70,80)-(125,109),M2
145 'Start animation
150 FOR X = 0 TO 360 STEP 5
160     PUT(74+X,80),M2,PSET
170     FOR TIME=0 TO 100: NEXT TIME
180     PUT(490,75),M1,XOR
190 NEXT X
200 LINE(450,50) - (480,80)
210 LINE(435,70) - (477,85)
240 END
250 DRAW "H1 L2 H2 L2 H2 U10 E2 R1 E5 R1 U2 H1 U1 H2"
260 DRAW "U2 E3 R10 F3 D2 G2 D1 G1 D2 R1 F5 R1 F2 D10 G2 L2 G2 L2 G1 L10 "
270 PAINT(X1+5,100),1
280 RETURN
```

〔그림 4〕 텍스트 모드에서의 페이지



한 화면을 구성하고 있는 것은 아니며, 한 화면을 디스플레이 하는데 알맞은 용량(페이지)으로 쪼개어 사용하고 있다. 그럼 이 16K바이트 용량으로 몇개의 화면 즉 몇개의 페이지를 만들어 낼 수 있을까? 결론부터 말하자면 80칼럼(WIDTH 80)에서는 0부터 3번까지 4개의 페이지가, 40칼럼(WIDTH 40)에서는 0부터 7까지 8개의 페이지가 사용될 수 있다.

화면에 문자 하나를 디스플레이 시키기 위해 메모리에서는 2바이트가 필요하다. 즉 한 바이트는 그 문자의 ASCII 코드값을 또 한 바이트는 그 문자의 칼라코드 값을 가진다. 그러므로 80칼럼의 모드에서는 4000바이트의 메모리가 필요하게 된다.

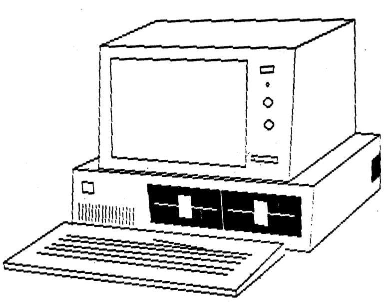
(80문자) * (25줄) * (2바이트/1문자) = 4000바이트

하나의 화면이 4000바이트이므로 16K바이트로는 4개의 화면을 만들고도 남는 용량이 조금 생긴다. 즉 하나의 페이지에 4K바이트(4096바이트)를 할당하고 있어서 마지막의 96바이트는 쓰이지 않고 있다. 마찬가지 방법으로 40칼럼 모드에서는 하나의 화면당 2000바이트가 할당되어 8개의 페이지를 사용할 수 있다(그림 4 참조).

리스트 8은 텍스트모드에서의 애니메이션을 보인 것으로, 두개의 페이지를 사용하여 한 페이지에는 왼쪽 발을 올린 이미지, 또다른 페이지에는 오른쪽 발을 올린 이미지를 반복시킴으로써 걸어가고 있는 효과를 내었다.

〔리스트 8〕 텍스트 모드에서의 애니메이션

```
1 'ANIMATE.BAS program
5 '----------------------------------------------
10 SCREEN 0,1,0:KEY OFF: CLS
20 WIDTH 40: LOCATE ,,0:CLS:SCREEN ,,1: CLS
30 S$ = CHR$(32) + CHR$(32)
40 A$ = CHR$(8) + CHR$(8)
50 B$ = CHR$(219) + CHR$(219)
60 C$(0) = CHR$(188) + CHR$(186)
70 D$(0) = CHR$(32) + CHR$(200)
80 C$(1) = CHR$(186) + CHR$(200)
90 D$(1) = CHR$(188)
100 A = 2: B = 34: C = 1
110 '----------------------------------------------
120 FOR I = A TO B STEP C
130   GOSUB 220
140   FOR TIME = 1 TO 50: NEXT TIME
150 NEXT I
160 FOR L = 1 TO 30
170   GOSUB 220
180 NEXT L
190 SWAP A,B: C = -C
200 GOTO 120
210 '----------------------------------------------
220 SCREEN ,,J,1-J
230 LOCATE 5,I: PRINT S$ + A$ + S$
240 LOCATE 6,I: PRINT S$ + B$ + S$
250 LOCATE 7,I: PRINT S$ + C$(J) + S$
260 LOCATE 8,I: PRINT S$ + D$(J) + S$
270 J = 1 - J
280 IF INKEY$ <> "" THEN SCREEN 0,1,0,0: END
290 RETURN:'----------------------------------------
```



어셈블리의 기본 개념

박승제/여주컴퓨터학원장

어셈블리의 개요

포트란이나 코볼 등의 고급언어의 출현은 프로그램의 체계를 일시에 뒤바꾸어 놓는 일대의 변혁이요 혁명이었다. 당시에는 모든 프로그램을 어셈블리로 작성하였고, 그렇기 때문에 프로그래머는 어셈블리의 명령어보다는 하드웨어의 구조를 먼저 익혀야 했다. 하드웨어의 구조를 모르고 프로그램을 작성한다는 것은 어셈블리 프로그램에서는 불가능하기 때문이다. 그러나 고급언어가 등장하면서부터 컴퓨터의 구조를 모르고도 프로그램을 작성하는 것이 가능하게 되었다.

프로그래머는 자신이 사용하는 언어의 문법에 적절한 명령어만 기술하면 하드웨어적인 처리는 컴파일러가 알아서 하기 때문에 프로그램을 개발하는데 많은 시간과 경비를 줄일 수 있었다. 이것은 획기적인 것이었지만 그렇다고 고급언어가 프로그래머나 사용자의 욕구를 모두 충족시켜 줄 수 있는 것은 아니다.

8비트 시절과 비교해서 16비트 컴퓨터에는 더 많은 범용언어들이 각각의 분야에 활용되고 있지만 이 언어들의 장점으로 꼽히는 것들 중의 하나가 바로 '어셈블리와의 인터페이스'이고 보면 실로 아이러니칼한 일이 아닐 수 없다.

고급언어의 등장은 어셈블리를 추방할 것처럼 여겨졌고 대부분의 응용 프로그램은 하드웨어와는 무관하게 작성되며, 이것이 프로그램의 이식성을 좋게 하는 요인 중의 하나였다. 즉, 기종에 구애받지 않고 프로그램을 작성할 수가 있게 된 것이다.

그러나 고급언어는 어떤 한계가 있기 마련이고 그 한계를 극복하는 것은 어셈블리에서 가능하므로 고급언어에서는 그런 부분은 어셈블리로 작성하여 두고, 필요시 하나의 서브루틴으로 호출하여 실행하는 방법을 사용하게 된다. '컴퓨터를 사용하는 한 결코 어셈블리는 사라지지 않는다'고 하면 너무 극단적인 표현일까? 어쨌든 어셈블리는 아직도 쓰이고 있으며, 고급언어가 하지 못한 세세한 부분은 당연히 어셈블리의 영역으로 되어 있다.

어셈블리는 컴퓨터를 직접 제어하므로써 고급언어가 처리하지 못한 일을 가능케 한다. 또한 여타 고급언어로 작성된 프로그램보다는 훨씬 수행 속도가 빠르다.

그렇다고 고급언어에서 충분히 처리할 수 있는 것을 구태여 어셈블리로 작성한다는 것이 도리어 시간과 노력의 낭비이다. 따라서 어셈블리는 고급언어에서 하지 못하는 특수한 처리를 지원해 주는 하나의 서브루틴 정도로 활용하는 것이 어떨까 하는 생각이 든다. 아무튼 독자들이 고급언어를 사용하고 있다면, 또 자신이 사용하는 언어의 단점이나 한계를 극복하려고 생각한다면, 이 강좌를 주시하기 바란다.

8비트와 16비트의 차이

컴퓨터는 중앙처리장치라는 것이 있고 이것을 CPU(Central Process Unit)라고 한다는 것은 컴퓨터를 배우는 국민학생들도 다 아는 사실일 것이다.

CPU에는 레지스터(Register)라고 하는 임시 기억장소가 있는데, 이 레지스터들은 메모리에 저장되어 있는 데이터를 가공하거나 메모리에 있는 명령어들을 해석하고 실행하는 일을 담당하고 있다. 이 레지스터의 크기가 8비트인 CPU는 한번에 8비트(1바이트)의 자료를 처리할 수 있다. 또 이 레지스터의 크기가 16비트인 컴퓨터는 한번에 16비트(2바이트)의 자료를 처리할 수 있다.

8비트의 CPU가 2바이트가 자료를 처리하는 경우에는 8비트씩 2회에 하게 되는 것을 16비트는 단 1회에 처리하므로 이론적으로는 16비트 CPU의 처리속도가 8비트의 CPU에 비해 2배 빠르게 된다. 또한 8비트 CPU는 메모리 번지의 지정에 16비트를 할당하고 있으므로 2^{16}=65536=64KB의 메모리를 다룰 수 있지만, 16비트 CPU의 경우에는 주소의 지정을 20비트로 하고 있으므로 2^{20}=1MB의 메모리를 취급할 수 있게 된다.

CPU는 한번에 8비트 혹은 16비트의 자료를 처리하므로 8비트를 메모리에서 가져오려면 최소한 8개의 전선이 필요하다. 이 전선을 버스(BUS)라고 부르는데 데이터를 가져오는 버스를 데이터 버스(Data Bus), 주소를 지정하는 버스를 어드레스 버스(Address Bus), 제어 신호를 보내는 버스를 콘트롤 버스(Control Bus)라고 한다. 8

비트의 컴퓨터는 데이터 버스가 8가닥, 어드레스 버스가 16가닥으로 되어 있다. 그렇다면 16비트 컴퓨터는 데이터 버스가 16가닥, 어드레스 버스는 20가닥으로 되어 있어야 한다.

그러나 16비트 CPU라는 I8088(이하 8088이라 칭함)을 사용하는 IBM PC에는 데이터 버스가 8가닥, 어드레스 버스가 20가닥으로 되어 있다. 데이터 버스가 8가닥이므로 한 번에 16비트를 처리하는 레지스터를 가지고 있더라도 데이터를 8비트씩 2회에 걸쳐 전송하게 된다. 프로그램을 작성할 때는 한번에 16비트를 전송하는 것으로 하지만 실제 CPU가 동작할 때는 8비트씩 2회 전송하게 된다. 이것은 레지스터의 능력은 8비트의 2배이지만 데이터의 전송은 8비트에 비해 별로 우수하지 못한 셈이며, 16비트가 8비트에 비해 2배가 빠르다는 것이 단순히 이론일 따름이라는 결론에 도달하게 된다. (실제의 빠르기는 데이터 버스의 크기가 아니고 클럭 주파수라는 것으로 측정한다.)

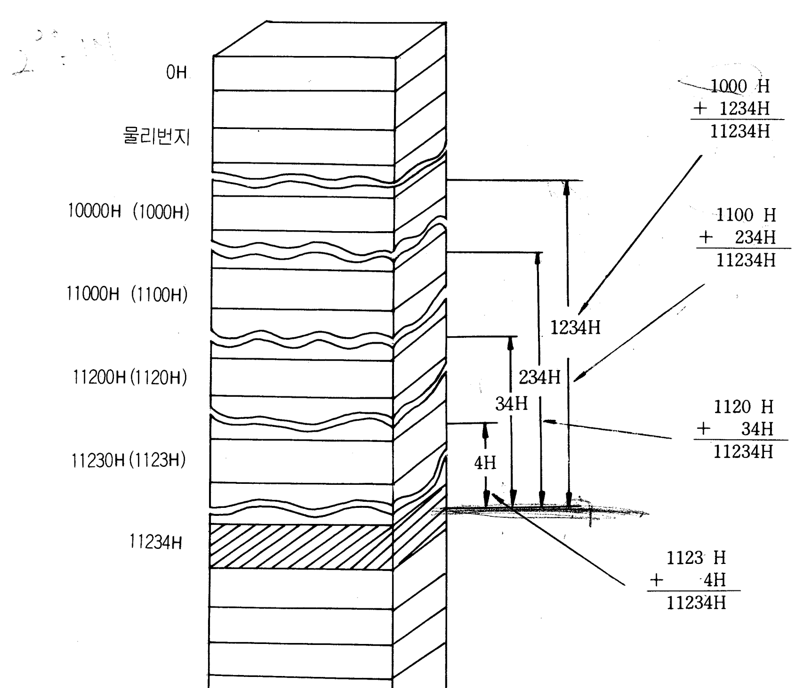
또 어드레스 버스가 20가닥이라고 하였으나 주소를 지정하는 인덱스 레지스터의 크기는 8비트 CPU와 같은 16비트의 크기이다. 그러므로 8088에서 다루는 주소의 크기 역시 64KB를 넘지 못한다. 8088에서는 이 64KB의 한계를 극복하기 위해 세그먼트(Segment) 라는 개념을 도입하였다.

세그먼트의 개념

16비트 레지스터로 나타낼 수 있는 값은 16진수로 0~0FFFFH의 값이다. 8088에서는 번지의 지정에 2개의 레지스터를 사용하며, 이 레지스터를 16진수로 1자리(2진수 4자리)만큼 비켜서 덧셈을 하는 방식으로 20비트의 주소를 표현한다. 예를 들면 다음과 같이 된다.

```
  1000 H
+ 1234H
-------
 11234H
```

〔그림 1〕 번지 지정의 예



*( )속의 수치는 세그먼트의 값

이때 상위 4자리(16진수)를 나타내는 값(앞의 예에서 1000H)을 세그먼트라고 하며, 하위 4자리를 나타내는 값(앞의 1234H)을 오프셋(Offset)이라고 한다.

이 세그먼트의 값은 실제로는 10000H의 번지를 가리키게 되며, 이 번지로부터 1234H가 떨어진 곳이 바로 오프셋이 갖는 의미이다. 즉, '어느 한 번지로부터 상대적으로 얼마 떨어진 번지'라는 식으로 주소를 지정하게 된다. 이것은 주소를 간접 지정하는 방법이므로 같은 번지라 하더라도 표현 방법이 수없이 많게 된다.

위에서 예로 든 11234H의 번지라도 다음과 같이 지정할 수가 있다(그림 1 참조). 따라서 번지를 지정하는 데 혼란을 가져올 수 있으므로 세그먼트의 지정은 '세그먼트 레지스터'를 지정하지 않고 오프셋만 지정하더라도 자동적으로 세그먼트 레지스터를 참조하여 번지를 계산하도록 되어 있다. 이 세그먼트는 0~0FFFFH사이면 어디든지 지정할 수 있으므로 20비트의 주소를 자유로이 나타낼 수 있다.

8088에서는 세그먼트를 지정하는 레지스터가 4개가 있다. 이 세그먼트는 레지스터는 CPU가 명령을 읽어내는 번지를 지정하는 레지스터(CS), 데이터를 가져오는 번지를 지정하는 레지스터(DS), 스택 동작을 수행하는 번지를 계산하는 레지스터(SS), 기타의 용도로쓰이는 레지스터(ES)가 있다.

이렇게 모든 번지에 세그먼트를 지정하게 되므로 장점보다는 단점이 더 많게 된다. 같은 번지라도 표현 방식이 다르다든지, 한번에 64KB이상의 데이터를 지정할 수 없다든지, 크기가 64KB 이상이 되는 프로그램을 작성하는 것이 곤란하다는 등의 문제가 바로 그것이다.

8088 CPU의 구성

8088에는 여러개의 레지스터가 있는

데 이들은 용도에 따라 범용 레지스터, 세그먼트 레지스터, 특수 레지스터로 구별되며, 각각 고유한 명칭이 있다(그림 2 참조). 이 장에서는 레지스터의 개략적인 기능에 대해 설명한다.

AX(Accumulator): 전송 및 연산에 사용된다. '누산기'라고도 불리우며, 곱셈, 나눗셈에는 반드시 AX(AL)을 이용하여야 하는 등 가장 빈번히 쓰이는 레지스터이다.

BX(Base Register): 전송 및 연산에 쓰인다. 번지의 간접 지정에 쓰이기도 한다.

CX(Counter Register): 전송 및 연산에 이용한다. 반복 실행 명령에 있어서 반복 횟수의 카운트에 쓰이기도 한다.

DX(Data Register): 전송 및 연산에 쓰인다. 곱셈, 나눗셈 등의 명령에서 AX 레지스터와 짝이 되어 쓰이기도 한다.

SP(Stack Pointer): 스택의 마지막 데이터 위치를 기억하는데 쓰인다.

BP(Base Pointer): 전송 및 연산에 쓰인다. 스택상의 데이터 번지를 간접 지정할 때 쓰이기도 한다.

SI(Source Index): 번지의 간접지정에 쓰인다. 메모리에서 레지스터로 전송할 때 메모리의 번지 지정에 쓰이기도 한다.

DI(Destination Index): 번지의 간접 지정에 쓰인다. 레지스터에서 메모리로 전송할 때 메모리의 번지를 지정하는데 쓰이기도 한다.

IP(Instruction Pointer): 프로그램을 실행할 때 다음에 실행할 명령이 저장되는 번지를 가리키는 프로그램 카운터로 쓰인다. 사용자가 그 값을 변경할 수는 없다.

FL(Flag Register): 연산이나 명령 실행후의 상태를 기억하는데 쓰인다. 이를테면 연산 후 0(zero)이 되었다든지, 자리 올림이 발생했는지의 상태 여부를 기억한다.

CS(Code Segment Register): CPU가 실행해야 할 명령이 저장되어 있는 세그먼트의 값을 가리킨다. CPU는 CS와 IP의 값을 계산하여 번지를 계산한 다음 그 곳의 명령을 실행하게 된다.

DS(Data Segment Register): 데이터가 저장된 세그먼트를 기억한다.

SS(Stack Segment Register): 스택이 위치하는 세그먼트를 기억한다.

ES(Extra Segment Register): 다른 세그먼트 내의 자료를 전송할 때 수신측의 세그먼트를 지정하는데 이용된다.

프로그램 작성시의 준비사항

어셈블리 프로그램을 작성하기 위해서는 에디터와 어셈블러, 링커, 디버거 등이 필요하다. 에디터는 DOS 디스켓에 수록되어 있는 EDLIN이나 SIDE KICK의 Notepad, Edix등 Text를 ASCII 형식으로 디스켓에 저장하는 에디터면 된다. 어셈블러는 쉽게 구할 수 있으며 주로 MASM(매크로 어셈블러)을 이용하는데 요즈음은 터보 어셈블러를 사용하는 프로그래머들도 많다.

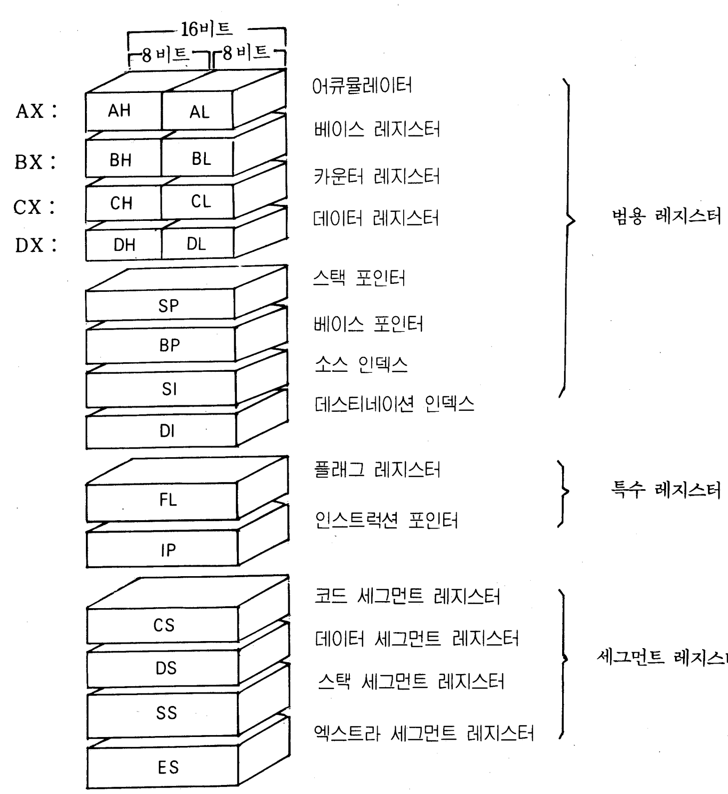
링커는 대부분의 어셈블러 디스켓에 함께 수록되어 있다. 디버거는 프로그램을 디버깅하는 데 쓰이며, DOS 디스켓의 DEBUG.COM을 이용하면 된다.

어셈블리 프로그램을 배우는 데는 어떤 어셈블러를 이용하든 상관이 없겠으나 이 강좌의 예제 프로그램으로 수록된 것은 모두 MASM으로 어셈블하였다.

프로그램의 기술 방법

어셈블리 프로그램은 행번호를 기술하지 않는다 어셈블리는 다음과 같은 형식으로 기술된다.

〔그림 2〕 8080 레지스터의 구성



레이블 : OP코드 제1 오퍼랜드, 제2 오퍼랜드 ;주석문

레이블과 주석문은 필요시에만 기술하며, 명령에 따라서는 오퍼랜드가 하나이거나 오퍼랜드가 없는 경우도 있다. OP코드(Operation Code)는 구체적인 동작을 지시하며, 오퍼랜드는 명령의 대상이 되는 레지스터나 자료이다.

이때 조작의 방향은 제2 오퍼랜드에서 제1 오퍼랜드로 이루어진다. 즉, MOV AX, BX의 명령은 BX 레지스터의 내용을 AX 레지스터로 이동(Move)하라는 명령이다.

어셈블리는 명령을 행단위로 하고 있다. 따라서 도중에 줄을 바꾸면 안된다. 단, 레이블과 주석문은 한 줄에 하나씩 기술할 수도 있다. 즉, 다음의 두 경우는 같다.

```
L1: MOV AX, BX ;BX의 내용을 AX로 옮김

L1:
   MOV AX, BX
  ;BX의 내용을 AX로 옮김
```

';'은 그 행이 주석문임을 어셈블러에게 인식시키는 식별자이다.

어셈블러는 영어의 대문자나 소문자를 구별하지 않기 때문에 대소문자를 적절히 섞어서 프로그램을 보기 쉽게 작성할 수도 있다. 또한 1개 이상의 공백은 어셈블러가 무시하므로 명령어사이에 적절한 공백을 부여하여 프로그램을 알아보기 쉽게 작성하는 것이 좋다. 보기 1보다는 보기 2의 프로그램이 이해하기 쉽고 보기에도 좋다.

〔보기 1〕

```
SAMPLE SEGMENT
ASSUME CS:SAMPLE
MOV DL,'A'
MOV AH,2
INT 21H
MOV DL,42H
INT 21H
MOV BL,'C'
MOV DL,BL
INT 21H
MOV AH,4CH
INT 21H
SAMPLE ENDS
END
```

〔보기 2〕

```
SAMPLE  SEGMENT
        ASSUME  CS:SAMPLE
        MOV   DL,'A'
        MOV   AH,2
        INT   21H
        MOV   DL,42H
        INT   21H
        MOV   BL,'C'
        MOV   DL,BL
        INT   21H
        MOV   AH.4CH
        INT   21H
SAMPLE  ENDS
        END
```

소스프로그램의 형식

어셈블리 프로그램은 다음과 같은 형식으로 기술한다.

```
<세그먼트 이름> SEGMENT
        ASSUME CS : <세그먼트 이름>

        ┌──────────────┐
        │  프로그램 본체  │
        └──────────────┘

<세그먼트 이름> ENDS
        END
```

〈세그먼트 이름〉은 사용자가 정해준 단어이며, 31자 이내의 영문자와 숫자, 그리고 '_'(Under Score)로 구성되는데 반드시 첫문자는 영문자이어야 한다.

위의 형식 중 SEGMENT, ASSUME, ENDS, END 등은 실제로 기계어를 생성하는 것이 아니고, 어셈블러가 프로그램의 본체를 번역할 때 참조하게 된다. 이와 같이 기계어를 생성하지 않는 명령을 어셈블리에서는 의사 명령(Pseudo Instruction)이라고 한다.

백문이 불여 일견이라고 실제로 예제 1과 같은 프로그램을 작성해 가면서 형식을 익혀보기로 한다.

〈예제 1〉 화면에 'ABC'를 출력한다.

```
1  SAMPLE  SEGMENT
2          ASSUME  CS:SAMPLE
3          MOV     DL,'A'
4          MOV     AH,2
5          INT  21H
6          MOV     DL,42H
7          INT  21H
8          MOV     BL,'C'
9          MOV     DL,BL
10         INT  21H
11         MOV     AH,4CH
12         INT  21H
13 SAMPLE  ENDS
14         END
```

어셈블리에서는 행번호를 기술하지 않지만 위의 예제에서는 설명의 편의를 위해 행번호를 붙여둔 것이므로 독자들이 입력할 때는 행번호는 입력하지 않는다. 이 프로그램을 행단위로 번역하면 다음과 같다.

1 : 이 세그먼트의 이름은 SAMPLE이다.(선언부)
2 : CS 레지스터는 SAMPLE이 시작하는 번지와 일치시켜라.(선언부)
3 : DL 레지스터에 'A'의 아스키 코드값(41H)을 기억시켜라.
4 : AH 레지스터에 2를 기억시켜라.
5 : DOS의 인터럽트 21H를 호출하라.
6 : DL 레지스터에 16진수 42H를 기억시켜라.
7 : DOS의 인터럽트 21H를 호출하라.
8 : BL 레지스터에 'C'의 아스키 코드값(43H)를 기억시켜라.
9 : BL 레지스터에의 내용을 DL 레지스터로 이동하라.
10 : DOS 인터럽트 21H를 호출하라.
11 : AH 레지스터에 16진수 4CH를 기

억시켜라.

12 : DOS 인터럽트 21H를 호출하라.

13 : SAMPLE 세그먼트는 여기서 끝난다.

14 : 프로그램의 끝이다.

위의 프로그램에서 보면 PRINT나 WRITE등 고급언어에서 쓰는 명령어는 하나도 없고 MOV와 INT 21H만이 잔뜩 나열되어 있다. 이와 같이 어셈블리 프로그램에서는 프로그래머 자신이 레지스터를 직접 조작하여 어떤 처리결과를 유도하게 된다. 이것이 어셈블리가 고급언어에 비해 어려운 점의 하나이다.

프로그램의 자세한 설명은 뒤로 미루고 우선 이 프로그램을 어셈블하고 실행하여 보기로 한다.

먼저 이 프로그램을 SAMPLE.ASM이라는 이름으로 디스크에 저장한 다음 DOS로 복귀한다. 매크로 어셈블러는 어셈블시 확장자를 지정하지 않으면 자동적으로 .ASM으로 하고 있으므로 소스 프로그램의 확장자를 .ASM으로 해두는 것이 좋다.

DOS에서 방금 저장한 디스켓을 B: 드라이브에 넣고, A: 드라이브에는 어셈블러 디스켓을 넣는다. 그런후 A>MASM을 입력한다. 그러면 화면 1 과 같은 메세지가 출력되며, 화면에서 보는 바와 같이 소스 화일명과 오브젝트 화일명을 묻게 된다.

이때 에러가 발생하면 다시 에디터를 호출하여 SAMPLE.ASM을 로드한 다음 프로그램을 수정하여 디스크에 저장하고 위의 과정을 반복한다.

디스크의 디렉토리를 보면(화면 1 참조) SAMPLE.OBJ와 SAMPLE.LST 가 생성되어 있음을 알 수 있는데, SAMPLE.OBJ가 기계어 모듈이다. 이 기계어 모듈은 실행 개시번지 등을 가지고 있지 않기 때문에 실행할 수 없으며, 링커로 하여금 번지등을 부여하도록 해야만 한다.

A>LINK 을 입력하면 화면 2와 같은

〔화면 1〕 어셈블하는 과정

```
A:\>MASM
Microsoft (R) Macro Assembler  Version 4.00
Copyright (C) Microsoft Corp 1981, 1983, 1984, 1985. All rights reserved.

Source filename [.ASM]: B:SAMPLE
Object filename [SAMPLE.OBJ]:
Source listing  [NUL.LST]: B: SAMPLE
Cross-reference [NUL.CRF]:

  50516 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

A:\>
A:\>DIR B:

Volume in drive B has no label
Directory of B:\

SAMPLE   LST   1477   12-29-88    1:33a
SAMPLE   OBJ     60   12-29-88    1:33a
SAMPLE   ASM    295   12-29-88   12:15a
        3 File(s)    123392 bytes free
A>
```

〔화면 2〕 링크과정과 프로그램의 실행

```
A:\>LINK

Microsoft (R) 8086 Object Linker  Version 3.05
Copyright (C) Microsoft Corp 1983, 1984, 1985.  All rights reserved.

Object Modules [.OBJ]: B:SAMPLE
Run File [SAMPLE.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: no stack segment
A:\>DIR B:

Volume in drive B has no label
Directory of B:\

SAMPLE   LST     1477   12-29-88    1:33a
SAMPLE   OBJ       60   12-29-88    1:33a
SAMPLE   EXE      532   12-29-88   12:15a
SAMPLE   ASM      295   12-29-88   12:15a
        4 File(s)  122368 bytes free

A>B:SAMPLE
ABC
A>
```

메시지가 나오므로 화일이름을 적당히 입력한다. 이때 Warning:no stack segment 라는 경고에러가 나오는데 이것은 무시하도록 한다.

디렉토리에는 SAMPLE.EXE가 생성되는데 이것이 바로 DOS에서 실행이 가능한 화일이다.

프로그램의 실행은 DOS상에서 프로그램의 이름을 입력하면 된다. A>B:SAMPLE을 입력하므로써 화면에는 ABC가 출력될 것이다.

이러한 일련의 절차들은 컴파일러형의 언어들을 사용해보지 않는 독자라면 다소 번거로운 것이지만 대부분의 컴파일러형의 언어(FORTRAN, COBOL, PASCAL…)에서는 이러한 절차를 밟아서 확장자가 .EXE인 화일을 생성하게 된다.

위에서 .LST인 화일은 소스 프로그램을 기계어로 번역한 결과를 수록해둔 화일로서 그 내용은 **화면 3**과 같다.

화면 3에서 보면 소스 프로그램과 1:1로 대응해서 기계어 코드를 생성시키는 것을 알 수 있으며, 또한 기계어 코드가 없는 곳이 눈에 띄기도 한다. 이 기계어 코드가 없는 부분이 바로 의사 명령어로 작성된 부분이다.

이후부터는 소스 프로그램에 행번호를 부여하여 프로그램을 설명하는 것이 아니라 리스팅 화일의 번지를 가지고 프로그램을 설명하게 된다. 즉, 0004번지의 내용은 어떠어떠한 것이다라는 식이 될 것이다.

〔화면 3〕 리스팅 화일의 내용

```
Microsoft (R) Macro Assembler  Version 4.00              1/1/80 00:07:08

                                                         Page     1-1


0000                          SAMPLE      SEGMENT
                                          ASSUME    CS:SAMPLE
0000  B2 41                               MOV       DL,'A'
0002  B4 02                               MOV       AH,2
0004  CD 21                               INT 21H
0006  B2 42                               MOV       DL,42H
0008  CD 21                               INT 21H
000A  B3 43                               MOV       BL,67
000C  8A D3                               MOV       DL,BL
000E  CD 21                               INT 21H
0010  B4 4C                               MOV       AH,4CH
0012  CD 21                               INT 21H
0014                          SAMPLE      ENDS
                                          END

Microsoft (R) Macro Assembler  Version 4.00              1/1/80 00:07:08

                                                         Symbols-1


Segments and Groups:

                 N a m e                Size    Align    Combine Class

SAMPLE . . . . . . . . . . . . . .      0014    PARA     NONE

      14 Source  Lines
      14 Total   Lines
      22 Symbols

   50516 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors
```

데이터의 전송명령

어셈블리 프로그램을 작성하기 위해서는 먼저 8088의 레지스터의 구조와 기능을 이해하지 않으면 안된다.

이 레지스터들은 8비트 크기와 16비트 크기가 있는데 데이터를 전송할 경우에는 전송하려는 레지스터나 자료의 크기와 전송받는 레지스터의 크기가 일치해야 한다.

데이터 전송 명령(MOV)는 다음과 같은 경우에 이용한다.

- 레지스터에 수치를 대입한다.
- 레지스터의 내용을 레지스터로 이동한다.
- 레지스터의 내용을 메모리로 이동한다.
- 메모리의 내용을 레지스터로 이동한다.

메모리에서 메모리로 데이터를 전송하는 명령은 없으므로 이 경우에는 메모리의 내용을 레지스터로 일단 옮긴 다음 레지스터의 값을 메모리로 전송하는 방법을 이용하게 된다.

메모리의 내용을 레지스터로 이동하는 명령을 Load, 레지스터의 내용을 메모리로 이동하는 명령을 Store라고 기술하는 CPU도 있으나 8088에서는 양쪽 모두 MOV(Move)라는 명령을 쓴다.

그림 2에서 보면 AX, BX, CX, DX

등의 레지스터가 있는데 이들의 크기는 모두 16비트이다. 이들은 다시 둘로 나누어 8비트의 데이터를 다룰 수 있도록 하였는데 이들은 각각 독립된 레지스터로 활용할 수 있다.

이들 레지스터에 수치를 대입할 경우에는 2진수, 8진수, 10진수, 16진수 어느 것이나 사용할 수 있는데 수치의 표기에는 다음과 같은 규칙이 있다.

- 10진수 : 수치의 끝에 D를 붙인다. (D는 생략할 수 있고 또 흔히 생략한다.)
- 2진수 : 수치의 끝에 B를 붙인다.
- 8진수 : 수치의 끝에 O를 붙인다.(O는 알파벳이다.)
- 16진수 : 수치의 끝에 H를 붙인다. 수치가 A~F로 시작하는 경우에는 수치의 선두에 0을 붙여서 레이블과 구분한다.(0은 제로이다)

시스템 콜(System Call)

고급 언어에서는 PRINT나 WRITE에 의해 화면에 계산 결과나 문자열이 출력된다. 그러나 컴퓨터의 입장에서 보면 문자열을 출력하기 위해서 다음과 같은 일련의 명령이 수행된다.

(1) 현재 커서의 위치를 계산한다.
(2) (1)의 위치에 해당하는 스크린 버퍼에 그 문자의 아스키 코드를 넣는다.
(3) 문자열이 끝이 아니면 (1)~(2)를 반복한다.
(4) 모니터를 연결하는 입출력 포트(I / O Port)를 지정한다.
(5) 스크린 버퍼의 내용을 I / O 포트에 출력한다.
(6) 모니터를 제어하는 LSI는 (5)의 내용을 화면에 표시한다.
(7) 문자열의 표시가 끝나지 않았으면 (5)~(6)을 반복한다.

이런 일련의 절차들은 기계어 명령으로 수십, 혹은 수백 바이트에 이르게 된다.

프로그래머가 이를 일일이 지정하여 프로그램을 작성한다고 하면 이것은 너무나도 큰 일이 아닐 수 없다.

이렇듯 하드웨어에 밀접한 부분은 컴퓨터의 제작회사(운영체계의 설계회사)에서 기계어 서브루틴으로 미리 제작하여 두고, 필요시 그 부분을 호출(Call)하는 것 만으로도 그런 일련의 처리가 가능하게 하였다.

이와 같은 처리 루틴은 IBM-PC의 경우 BIOS(Basic Input Output System)와 DOS 인터럽트의 2가지가 있다. 대부분의 처리에 DOS인터럽트 루틴을 호출하는 것 만으로도 충분하며, 이 DOS 인터럽트 루틴에서 BIOS를 호출하는 경우가 많으므로 여기서는 BIOS는 거론하지 않기로 한다.

DOS 인터럽트의 루틴은 그 종류가 많고 다양하지만 가장 기본적인 처리를 하는 것이 바로 21H 인터럽트이다. 이것을 흔히 DOS Function Call이라고 부른다.

도스의 펑션 콜은 시스템콜이라고도 부르며, 어셈블리에서는 INT 21H로 실행하게 된다.

시스템 콜은 70여 가지의 내부 처리 루틴을 가지고 있으며 각각의 루틴에 번호가 매겨져 있다.

이 번호를 AH 레지스터에 넣고 INT 21H를 실행하면 원하는 처리를 할 수가 있게 된다.

INT 21H가 실행될 때 다른 레지스터의 값을 참조하는 경우가 있는데 이것은 처리 루틴마다 다르다.

화면에 1문자 출력

Systen Call의 2번에 해당하는 것으로 AH 레지스터에 2를, DL 레지스터에 출력할 문자의 아스키 코드를 지정한 다음 INT 21H를 실행하면 된다.

문자나 1바이트의 수치를 지정하는 방법은 여러가지로 표현할 수 있으며 다음의 보기는 모두 동일한 결과가 나타난다.

```
MOV  DL,  41H
MOV  DL,  'A'
MOV  DL,  65
MOV  DL,  'C'-2
MOV  DL,  62+3
```

```
SAMPLE   SEGMENT
         ASSUME   CS:SAMPLE
         MOV      DL,'A'
         MOV      AH,2
         INT 21H
         MOV      DL,42H
         INT 21H
         MOV      BL,67
         MOV      DL,BL
         INT 21H
         MOV      AH,4CH
         INT 21H
SAMPLE   ENDS
         END
```

위의 샘플 프로그램(SAMPLE.ASM)을 살펴보면 시스템 콜을 호출한 부분이 3군데 있음을 알 수 있다.

프로그램의 종료

프로그램의 종료는 시스템 콜의 4CH를 호출함으로 종료된다. 즉,

```
MOV  AH, 4CH
INT  21H
```

이다. 이것은 매우 중요하므로 공식처럼 익혀두어야 한다. 물론 위의 시스템 호출이 없더라도 더 이상의 명령이 없으면 프로그램이 중단되겠지만 시스템 호출이 없이 중단되면 DOS로 돌아오지 않고 대부분 폭주해 버린다.

1문자의 입력

키보드에서 1문자를 입력하는 경우는 2가지 방법이 있다. 입력된 문자를 화면에 표시하느냐 하지 않느냐하는 2가지이다. 입력된 문자를 화면에 표시하는 것을 에코백(Echo Back)한다고 하며 이것은 시스템 콜의 1번으로 가능하며, 에코 백을 하지 않을 경우에는

〔리스트 1〕 2개의 문자를 입력받아 출력하는 프로그램

```
0000                 LIST    SEGMENT
                             ASSUME  CS:LIST
0000  B4 01                  MOV     AH,1
0002  CD 21                  INT 21H
0004  8A D8                  MOV     BL,AL
0006  B4 01                  MOV     AH,1
0008  CD 21                  INT 21H
000A  8A F8                  MOV     BH,AL
000C  B4 02                  MOV     AH,2
000E  B2 0D                  MOV     DL,0DH
0010  CD 21                  INT 21H
0012  B2 0A                  MOV     DL,0AH
0014  CD 21                  INT 21H
0016  8A D3                  MOV     DL,BL
0018  CD 21                  INT 21H
001A  8A D7                  MOV     DL,BH
001C  CD 21                  INT 21H
001E  B4 4C                  MOV     AH,4CH
0020  CD 21                  INT 21H
0022                 LIST    ENDS
                             END
```

시스템 콜의 8번을 이용한다.

어느 경우든 AH 레지스터에 호출번호를 넣고 INT 21H를 호출한다. 그러면 컴퓨터는 키보드로부터의 입력을 기다리다 입력이 되면 입력한 문자의 아스키코드를 AL 레지스터에 저장하여 준다.

그러면 예제 프로그램을 통해 1문자를 입력하는 방법을 익혀보자.

〈예제 2〉

2개의 문자를 입력받아 줄을 바꾼 후 출력한다.

처리 방법

시스템 콜의 1번으로 문자를 입력받는다. 이때 입력받은 문자의 아스키코드가 AL에 저장될 것이다.

이것을 BL에 잠시 대피하여 둔다. 다시 2번째의 문자를 입력받는다.

이때도 입력받은 문자가 AL에 저장될 것이다.

이것은 BH에 대피한다. 다음은 줄을 바꾼다. 줄은 0DH와 0AH를 연속해서 출력하면 바뀌게 된다.

다음은 BL과 BH의 내용을 차례대로 출력하면 된다.

이것을 프로그램으로 작성한 것이 리스트 1이다.

각자 입력하여 실행해 보기 바란다.

● 프로그램 설명

0000–0002 : 1문자 입력
0004 : 입력된 문자를 BL에 대피
0006–0008 : 1문자 입력
000A : 입력된 문자를 BH에 대피
000C–0014 : 화면상의 줄을 바꿈
0016–0018 : 처음에 입력한 문자의 화면 출력
001A–001C : 2번째 입력한 문자의 화면 출력
001E–0020 : 프로그램의 종료

위에서 보듯이 실제로는 별것도 아닌 프로그램이지만 그 내용은 실로 장황하기 그지없다.

고급 언어에서는 입출력 루틴을 컴파일러가 다 알아서 하지만 어셈블리에서는 모든 처리를 프로그래머가 해야 하기 때문이다.

그러나 어셈블리를 익혀두면 고급 언어에서는 도저히 흉내내지 못할 고도의 테크닉이나 기발한 처리가 가능하게 된다.

이제 어셈블리 프로그램을 어떻게 작성해야 하는가를 대강이나마 짐작하였을 것이다. 다음 호에는 문자열의 처리와 의사명령어를 사용하는 법, 데이터를 두는 법에 대해서 다루어 볼 예정이다. PC

데이터 정렬

박성현/디베이스 프로그래머

지난호에서는 데이터베이스 화일의 작성과 이를 어떻게 이용하는가에 대해 알아 보았다. 이번에는 데이터베이스 화일을 좀더 효율적으로 이용하여 더욱 효과적으로 사용할 수 있도록 익혀 보자.

데이터의 정렬

데이터를 입력하여 원하는 순서대로 정렬하지 못한다면 데이터가 많아 질수록 그 데이터는 데이터로서의 가치를 잃어 버리게 될 것이다. 데이터의 정렬이란 데이터를 관리하는데 있어 필수 요건이라 하겠다.

데이터를 정렬하는 방법은 두가지가 있다. 첫째는 데이터베이스 화일을 정렬시켜 새로운 데이터베이스 화일을 만드는 방법이고, 다른 방법으로는 인덱스화일과 함께 데이터베이스 화일을 열어 놓음으로써 인덱스 화일의 키포인트에 의해 마치 정렬된 것처럼 보이도록 하는 것이다.

첫째로 사용중인 데이터베이스 화일을 키필드에 의해 특정 순서대로 정렬을 하여 또 하나의 새로운 데이터베이스 화일을 만드는 방법을 알아보자.

```
. USE <database file-name>
```

위 명령으로 화일을 열어 놓고 다음의 명령을 사용한다.

```
. SORT ON <file-name> [/A] [/C] [/D] to
                                    <new-filename>
```

[/A] (Ascending)

오름차순에 의해 정렬하라는 옵션이다 (1, 2, 3, 4…A, B, C, D)

디폴트치로 정해져 있어서 옵션이 없을 경우 자연히 오름차순에 의해 정렬된다.

[/C]

이 옵션은 영어의 소문자와 대문자의 구별을 하지 말라는 옵션이다.

[/D] (Descending)

내림차순에 의해 정렬하라는 옵션이다 (Z, Y, X, W…9, 8, 7…)

/A와 /D는 동시에 사용할 수 없으며, /C를 /A 또는 /D와 동시 사용하기 위해서는 /AC 혹은 /DC로 사용해도 무방하다.

〈주의〉

위의 옵션을 사용하는데 있어서 한글 부분에서 약간의 문제점이 생긴다. 2바이트 조합형, 완성형의 경우는 한글영역이 아스키코드의 128번 이후(알파벳 소문자 이후)에 위치하므로 아무런 문제가 생기지 않는다.

그러나 7비트 완성형(청계천 한글, 옴니테크 영문 S/W용 한글등)의 경우는 한글영역이 알파벳 소문자와 같이 위치하고 있어 배열이 엉망이 되고 만다. 그러므로 이 한글카드를 사용하는 시스템에서는 한글과 같이 사용할 경우에는 반드시 알파벳은 대문자로 사용하고 위의 /C 옵션의 사용을 피해야 한다.

그럼 **리스트** 1에서 보는 바와 같이 지난호에서 사용했던 데이터베이스 화일을 이용하여 정렬하는 연습을 해보자.

다음은 인덱스 화일을 사용하여 데이터베이스 화일을 그대로 사용하면서 인덱스 화일을 함께 열어 정렬하는 방법을 알아보자.

인덱스는 소트와 매우 다른 방식을 취하고 있다. 이는 레코드들을 인위적으로 재구성하여 새로운 화일을 만드는 것이 아니라 데이터베이스 화일 자체는 입력된 순서 그대로 유지하면서 키에 따른 순서만을 설정해 놓은 화일을 새로 만든다. 이 화일은 실제의 데이터를 가지고 있지 않고 B-트리 구조에 의한 순서만을 갖고 있으므로 화일의 크기가 작아져 디스크장치를 효율적으로 사용하게 된다.

특히 어떤 레코드를 찾고자 할 때 매우 빠른 속도로 찾을 수 있게 된다. 또한 가장 중요한 것은 데이터베이스 화일과 인덱스 화일을 동시에 열어 놓고 데이터를 변경하면 인덱스 화일도 자동으로 변경되므로 재정렬은 생각하지 않아도 된다 (REINDEX라는 재정렬시키는 명령이 있지만 실수로 인덱스 화일을 열지 않고서 데이터를 변경했을 경우 외에는 거의 사용할 필요가 없다.)

특히 정렬은 하는데 있어 필드명으로만 키를 주는 것이 아니라 필드에 함수를 이용한 정렬이나 수식과의 계산 결과도 정렬이 가능하며 디베이스 사용자는 이 방법을 더 많이 사용한다.

다만 소트는 작은 화일중 데이터가

거의 바뀌지 않는 데이터베이스 화일만 소트하여 소트한 데이터베이스 화일을 사용한다 (데이터베이스 화일이 크면 데이터를 찾는데 유리한 인덱스를 사용하고 자주 내용이 바뀌는 것은 그때마다 정렬할 필요없는 인덱스를 사

〔리스트 1〕

```
. USE PC01              && PC01.dbf 라는 데이터베이스 화일을 열어 놓는다.

. LIST                  && 데이터베이스 화일의 내용을 본다.
Record# CODE   NAME         PAY  POINT BIRTH    CHECK
     1  000001 윤 택 중   850000  80.00 10/10/37 .T.
     2  000002 이 홍 룡   420000  65.50 07/08/61 .F.
     3  000003 박 준 영   340000  74.00 08/06/63 .T.
     4  000004 제 수 테   640000  85.50 12/27/58 .T.
     5  000005 김 규 봉   430000  80.00 12/18/62 .F.
     6  000006 최 영 숙   436000  74.00 05/13/66 .F.
     7  000007 김 현 진   614000  65.50 04/19/65 .T.
     8  000008 송 진 호   366500  65.50 07/31/60 .T.
     9  000009 최 현 호   366800  80.00 02/01/61 .F.
    10  000010 김 상 녕   807000  65.50 02/23/63 .T.
    11  000011 김 용 석  1240000  74.00 09/25/58 .F.
    12  000012 장 직 순   520000  80.00 06/24/56 .F.
    13  000013 이 정 민   690000  90.00 03/21/55 .F.
    14  000014 전 경 호   570500  90.00 11/20/56 .T.
    15  000015 김 동 균   492500  65.50 10/04/57 .T.

. SORT ON NAME TO PC01A     && name 필드를 올림차순으로 정렬하여 그 내용을
                            && PC01A.dbf를 새로 만들어 저장한다.
  100% Sorted          15 Records sorted

. USE PC01A                 && 새로 정렬된 화일을 연다.

. LIST                      && 정렬된 데이터베이스 화일의 내용을 본다.
Record# CODE   NAME         PAY  POINT BIRTH    CHECK
     1  000005 김 규 봉   430000  80.00 12/18/62 .F.
     2  000015 김 동 균   492500  65.50 10/04/57 .T.
     3  000010 김 상 녕   807000  65.50 02/23/63 .T.
     4  000011 김 용 석  1240000  74.00 09/25/58 .F.
     5  000007 김 현 진   614000  65.50 04/19/65 .T.
     6  000003 박 준 영   340000  74.00 08/06/63 .T.
     7  000008 송 진 호   366500  65.50 07/31/60 .T.
     8  000001 윤 택 중   850000  80.00 10/10/37 .T.
     9  000013 이 정 민   690000  90.00 03/21/55 .F.
    10  000002 이 홍 룡   420000  65.50 07/08/61 .F.
    11  000012 장 직 순   520000  80.00 06/24/56 .F.
    12  000014 전 경 호   570500  90.00 11/20/56 .T.
    13  000004 제 수 테   640000  85.50 12/27/58 .T.
    14  000006 최 영 숙   436000  74.00 05/13/66 .F.
    15  000009 최 현 호   366800  80.00 02/01/61 .F.

. USE PC01                  && 다시 PC01.dbf를 연다.

. SORT ON NAME /D TO PC01B  && name 필드를 내림차순으로 정렬하여 그 내용을
                            && PC01B.dbf 를 새로 만들어 저장한다.
  100% Sorted          15 Records sorted

. USE PC01B                 && 새로 정렬된 화일을 연다.

. LIST                      && 데이터베이스 화일의 내용을 본다.
Record# CODE   NAME         PAY  POINT BIRTH    CHECK
     1  000009 최 현 호   366800  80.00 02/01/61 .F.
     2  000006 최 영 숙   436000  74.00 05/13/66 .F.
     3  000004 제 수 테   640000  85.50 12/27/58 .T.
     4  000014 전 경 호   570500  90.00 11/20/56 .T.
     5  000012 장 직 순   520000  80.00 06/24/56 .F.
     6  000002 이 홍 룡   420000  65.50 07/08/61 .F.
     7  000013 이 정 민   690000  90.00 03/21/55 .F.
     8  000001 윤 택 중   850000  80.00 10/10/37 .T.
     9  000008 송 진 호   366500  65.50 07/31/60 .T.
    10  000003 박 준 영   340000  74.00 08/06/63 .T.
    11  000007 김 현 진   614000  65.50 04/19/65 .T.
    12  000011 김 용 석  1240000  74.00 09/25/58 .F.
    13  000010 김 상 녕   807000  65.50 02/23/63 .T.
    14  000015 김 동 균   492500  65.50 10/04/57 .T.
    15  000005 김 규 봉   430000  80.00 12/18/62 .F.

. SORT ON POINT /D , NAME TO PC01C  && point 필드를 내림차순으로 정렬후
                                    && 같은 내용은 name 필드를 올림차순으로
                                    && 정렬하여 그 내용을 PC01A.dbf 에
                                    && 저장한다.
  100% Sorted          15 Records sorted

. USE PC01C                 && 새로 정렬된 화일을 연다.

. LIST                      && 데이터베이스 화일의 내용을 본다.
Record# CODE   NAME         PAY  POINT BIRTH    CHECK
     1  000013 이 정 민   690000  90.00 03/21/55 .F.
     2  000014 전 경 호   570500  90.00 11/20/56 .T.
     3  000004 제 수 테   640000  85.50 12/27/58 .T.
     4  000005 김 규 봉   430000  80.00 12/18/62 .F.
     5  000001 윤 택 중   850000  80.00 10/10/37 .T.
     6  000012 장 직 순   520000  80.00 06/24/56 .F.
     7  000009 최 현 호   366800  80.00 02/01/61 .F.
     8  000011 김 용 석  1240000  74.00 09/25/58 .F.
     9  000003 박 준 영   340000  74.00 08/06/63 .T.
    10  000006 최 영 숙   436000  74.00 05/13/66 .F.
    11  000015 김 동 균   492500  65.50 10/04/57 .T.
    12  000010 김 상 녕   807000  65.50 02/23/63 .T.
    13  000007 김 현 진   614000  65.50 04/19/65 .T.
    14  000008 송 진 호   366500  65.50 07/31/60 .T.
    15  000002 이 홍 룡   420000  65.50 07/08/61 .F.
```

용한다.)

```
. USE <database filename>
```

위 명령으로 화일을 열어놓고 다음의 명령을 사용하여 인덱스 화일을 만든다.

```
. INDEX ON <key expression> TO <filename>
```

<key expression>

키 표현식은 문자, 숫자, 날짜 중 한가지 타입을 사용하여 여러개의 필드를 결합할 때는 한가지로 통일되어야만 한다. 그러므로 성명(Character)과 생년월일(Date)을 함께 인덱싱시키고자 할 때는 DTOC(birthday)을 사용하여 날짜형을 문자형으로 바꾸어 결합하여야만 한다.

소트와는 달리 모든 함수를 사용하여 결합할 수 있으므로 여러가지 함수를 사용하여 Ascending, Descending 등 원하는 형식을 100자까지 결합하여 Indexing을 할 수 있다.

<filename>

화일이름은 특별히 지정해 주지 않는한 생략치 드라이브와 .NDX가 확장자로 붙는다.

그럼 **리스트** 2와 같이 PC01.dbf를 이름순으로 정렬하는 인덱스 화일을 만들어 보자.

리스트 2에서 보는바와 같이 인덱스 화일을 만들어 데이터베이스 화일과 결합하면 레코드 번호는 원래의 번호를 유지하고 있음을 알 수 있다.

다음은 인덱스 화일을 만들때 특수한 방법들을 알아보자.

1) 내림차순으로 Indexing

문자필드의 경우는 ASC() 함수를 사용하여 아스키 코드 값으로 바꾼후 −(minus)를 사용하여 역수를 취한다. 숫자 필드의 경우는 -부호만 사용한다.

```
. INDEX ON -ASC(name) TO NAME01
. INDEX ON -point TO POINT01
```

〔리스트 2〕

```
. USE PC01

. INDEX ON NAME TO NAME01         && 이름 순서에 의한 인덱스 화일 NAME.ndx 를 만든다.
  100% indexed          15 Records indexed
. LIST                            && 인덱스 화일로 만들어진 인덱스 화일과 함께
                                     데이터를 읽는다.
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      5  000005 김 규 봉     430000  80.00 12/18/62 .F.
     15  000015 김 동 균     492500  65.50 10/04/57 .T.
     10  000010 김 상 녕     807000  65.50 02/23/63 .T.
     11  000011 김 용 석    1240000  74.00 09/25/58 .F.
      7  000007 김 현 진     614000  65.50 04/19/65 .T.
      3  000003 박 준 영     340000  74.00 08/06/63 .T.
      8  000008 송 진 호     366500  65.50 07/31/60 .T.
      1  000001 윤 텍 중     850000  80.00 10/10/37 .T.
     13  000013 이 정 민     690000  90.00 03/21/55 .F.
      2  000002 이 흥 룡     420000  65.50 07/08/61 .F.
     12  000012 장 직 순     520000  80.00 06/24/56 .F.
     14  000014 전 경 호     570500  90.00 11/20/56 .T.
      4  000004 채 수 태     640000  85.50 12/27/58 .T.
      6  000006 최 영 숙     436000  74.00 05/13/66 .F.
      9  000009 최 현 호     366800  80.00 02/01/61 .F.
```

2) 대소문자 구별하지 않고 Indexing

UPPER() 함수를 사용한다.

```
. INDEX ON UPPER(name) TO NAME01
```

3) 여러필드를 한꺼번에 Indexing

+(plus)를 사용하여 결합한다. 문자필드의 경우는(−)로도 연결이 가능하나(−)를 사용하면 자동적으로 TRIM() 함수를 사용한 것과 같이 뒤의 공백이 삭제되어 데이터의 길이에 따라 일관성이 없어지게 되므로 사용을 하지 않는다.

```
. INDEX ON SUBSTR(code,1,1)+name
          +DTOC(birth)+code TO CODE01
```

이때 숫자필드만으로 결합하는 경우, 수치를 계산하여 그 합계로 인덱스를 하므로 숫자 필드를 개별로 인덱스 시키고자 할 때는 다음과 같이 하여야 한다.

```
. INDEX ON  pay*1000 + point TO PAY01
```

또는

```
. INDEX ON STR(pay,7)+STR(age,5,2) TO PAY01
```

4) 데이터 필드를 연대순으로 Indexing

데이터 필드를 그대로 데이터필드로만 인덱싱시킬 때는 자동으로 연대순으로 인덱싱이 되지만 문자와 결합하여 DTOC() 함수를 함께 사용할 경우 (MM / DD / YY) 식으로 되어 연도보다 월일이 우선 순위가 된다. 이때는 간단히 'SET DATE ANSI'라는 명령으로 (MM / DD / YY)를 (YY / MM / DD) 형식으로바꾸어 사용하면 DTOC() 함수에 의해 변환해도 년월일의 순서대로 인덱싱을 할 수 있다.

반드시 (MM / DD / YY) 형식의 날짜를 사용하면서 문자와 함께 멀티 인덱싱을 해야 할 경우에는 다음과 같은 방법으로 하여 년월일 순으로 인덱싱을 할 수 있다.

```
. INDEX ON STR(YEAR(birth),4)
              + STR(MONTH(birth),2) ;
  + STR(DAY(birth),2) + name TO BIRTH01
```

여기까지 인덱스 화일을 만드는 방법을 알아 보았다. 정렬한 화일은 새로운 데이터베이스 화일 그 자체이므로 평상시 LIST, DISPLAY 등의 명령을 사용하면 그 순서에 따라 나오므로 다른 사용법이 있을 수 없다. 그러나 인덱싱을 했을 경우는 특수한 사용법에 의해 막강한 힘을 발휘하게 된다.

●인덱스 화일의 사용

1) 인덱스 화일의 열기

인덱스 화일은 단독으로 사용하지 못하고 반드시 데이터베이스 화일과 함께 사용하게 된다. 사용 방법은 데이터베이스 화일을 열때 함께 열어주면 된다(INDEX ON… TO…에 의해 인덱스 화일을 만든 그 시점에는 방금 인덱스 화일 하나가 열려 있는 상태이다).

```
. USE PC01  INDEX NAME01
```

2) 여러개의 인덱스 화일을 한꺼번에 열 수도 있다.

```
. USE PC01  INDEX NAME01, CODE01, BIRTH01
```

이때는 첫번째의 인덱스화일 순으로 정렬되어 있으며 나머지 인덱스화일은 아무 기능을 하지 못하고 있다. 그러나 APPEND나 EDIT 등으로 데이터가 변경되었을때 각자의 인덱스 화일 모두가 자동으로 재정렬되는 효과를 가진다. 이 첫번째 인덱스화일에 의해 데이터베이스 화일은 그 배열 순서대로 작동하게 되며 이것을 마스터 인덱스 화일이라 한다.

3) 마스터 인덱스화일은 수시로 바꿀 수 있다.

'SET ORDER TO 2'는 두번째의 인덱스화일을 마스터로 하라는 명령이다. 이렇게 되면 순식간에 데이터베이스 화일 구조는 CODE01.ndx의 화일 순으로 정렬되어 있는 모습을 볼 것이다. 첫번째 인덱스화일도 아직 열려 있으므로 항상 재정렬될 준비가 되어 있으며 다시 첫번째의 인덱스 화일을 마스터로 지정하고 싶으면 'SET ORDER TO 1'이라 명령하면 된다.

4) 데이터 베이스 화일을 열때 인덱스 화일을 함께 열지 않았다면 다음과 같이 인덱스 화일만 추가로 열 수도 있다.

```
SET INDEX TO NAME01,CODE01,BIRTH01.PAY01
```

이 명령은 하나만 더 추가할 수 없고 항상 같이 전체를 함께 설정해 주어야 한다.

5) 인덱스 화일을 열지 않고 데이터베이스 화일을 정정하였다면?

자동 업데이트가 되지 않았기 때문에 잘못된 정보를 제공하게 된다. 이때는 그 인덱스 화일을 열어주고 'REINDEX' 란 간단한 명령으로 재정렬 시킬 수 있다. 이런 실수를 방지하기 위해 항상 모든 인덱스 화일을 같이 열어서 'SET ORDER TO 〈number〉'로 마스터 화일을 변경해 가며 사용하며 REINDEX란 명령은 되도록이면 사용하지 않길 바란다. REINDEX 명령은 새롭게 인덱스 화일을 만드는 것과 같이 시간이 걸리는 명령이다.

6) 인덱스된 화일에서는 데이터를 찾는 시간이 무척 빨라진다.

보통의 데이터베이스 화일에서는 특정의 데이터를 찾는데 있어 화일의 처음부터 비교하여 찾기 때문에 다행이 화일의 윗 부분에 있으면 빨리 찾을 수 있지만 화일의 뒤에 존재한다면 데이터가 많아 질수록 그 데이터를 찾는 시간은 점점 길어 질 것이다. 그러나 인덱스 화일의 구조는 B-트리 구조를 가지고 있어 아무리 데이터가 많다 하더라도 순식간에 찾아낸다.

```
. USE PC01 INDEX NAME01
. LOCATE FOR NAME="김 동 균"
```

과

```
. SEEK "김 동 균"
```

의 속도를 비교해보라. 위와 같은 명령으로

```
. FIND  김 동 균
```

의 명령으로도 찾을 수 있다. FIND와 SEEK의 차이는 인용부호를 사용하지 않는 것과 사용하는 차이이다. 그러므로 SEEK로 찾을 때는 변수를 주어서 찾을 수가 있다. (FIND나 SEEK 명령은 데이터를 찾을때 빈번히 사용함으로 꼭 숙지하기를 바란다.)

이때 찾고자하는 레코드가 없을 경우는 현재화일의 맨 마지막에 위치해 있는다.

```
. FOUND ()

. RECNO ()
```

라고 명령하면 찾았을 경우에는 FOUND()는 .T. RECNO()는 찾은 레코드 번호를 가르쳐 준다. 그러나 찾지 못하는 경우에는 FOUND()는 .F.이고 RECNO()는 EOF()인 16이 나올 것이다.

7) 인덱스 화일의 닫기

사용중인 데이터베이스 화일을 닫으면 함께 열려있던 인덱스 화일도 함께 닫힌다.

```
CLEAR ALL , CLOSE ALL , CLOSE DATABASE , USE , QUIT
```

데이터베이스 화일은 열어 놓은 채로 인덱스 화일만 닫는다.

```
. CLOSE INDEX , SET INDEX TO
```

8) 인덱스 화일을 닫지 않은 채로 인덱스 화일을 사용하지 않는 것처럼 사용할 수도 있다.

```
. SET ORDER TO 0
```

이상으로 화일 정렬에 대해 자세히 알아 보았다. 어느 방법으로 사용하든 편리한 방법을 사용하여야 할 것이다. 데이터를 수정할 필요가 거의 없고 작은 화일의 경우는 구태여 인덱스 화일까지 만들 필요없이 정렬된 채로 유지하고 있으면 검색하는 시간이 길지 않으므로 오히려 편리할 것이다. 그러나 수정을 자주 해야하거나 데이터의 양이 많다면 인덱스를 사용하여야 좀 더

훌륭한 작동을 기대할 수 있을 것이다.

복수개의 화일사용

dBASE III PLUS에서는 데이터베이스 화일을 10개까지 동시에 열어 놓고 사용함으로써 데이터베이스 화일을 자주 열었다 닫는 불편함과 그에 따른 액세스 시간을 줄일 수가 있다. 1–10번까지의 작업영역에 각 1개의 데이터베이스 화일을 열어 놓을 수가 있다.

작업 영역을 설정하지 않고 사용하면 디베이스는 항상 작업영역 1에서 작업을 하게 된다.

작업영역 2에서 작업을 하고자 할 때는

```
SELECT 2
```

라는 명령만 하면 작업영역 2로 넘어가 그곳에서 또다른 데이터베이스 화일을 열고 작업을 할 수 있다.

지난호에서 사용하던 데이터베이스 화일에 다음의 데이터베이스 화일을 사용하여 실습해 보도록 하자.

```
Number of data records:      10
Date of last update    : 12/10/88
Field  Field Name Type        Width
    1  CODE       Character       6
    2  TEL        Character      10
    3  HMAIL      Character      10
** Total **                      27
```

PC02.dbf

위와 같은 데이터베이스 화일을 작성하였다면 우선 아래 **리스트 3**과 같은 순서대로 명령하여 데이터를 추가시킨다.

데이터의 추가가 끝났으면 아래의 **리스트 4**와 같은 순서대로 입력해 보길 바란다.

화일의 연결

앞에서는 1번 작업영역에서 1번과 2번의 데이터를 읽었을때 1번 작업영역의 것은 다 읽을 수 있지만 2번 작업영역의 것은 레코드가 이동되지 않기 때문에 현재 위치하고 있는 데이터만 반복해서 나왔다.

작업영역 1번의 데이터베이스 화일이 이동할 때 2번의 데이터베이스 화일도 따라 움직이게 하는 방법이 없을까?

이는 set relation to에 의해 데이터베이스 화일을 연결 시킴으로서 가능하다.

```
SET RELATION TO < 공통된 필드명 / RECNO() /
수치표현식 > INTO < 따라 움직이게될 데이타베이스 화일이
열려있는 작업영역의 알리아스(별명) >
```

〔리스트 3〕

```
. USE PC02              && 새로 만든 데이터베이스 화일을 연다.
. APPEND FROM PC01      && PC02.dbf 에 PC01.dbf의 데이터를 모두 추가 시킨다.
                           여기서는 필드가 같은 code 만 추가 되므로 다른 필드는
. BROWSE                && 직접 데이터를 입력 시킨다.
```

〔리스트 4〕

```
. select 1      &&---- 1 번 작업 영역에  PC01.dbf 를 열어 놓는다.
. use   PC01
. select 2      &&---- 2 번 작업 영역에  PC02.dbf 를 열어 놓는다.
. use   PC02
. select a      &&---- 다시 1 번 작업 영역에 돌아와 데이터를 읽는다.
                       select a 와 select 1 은 같은 명령이다. 숫자를
                       사용하여도 되고 알파벳을  사용하여도 무방하다.
. list
Record# CODE   NAME           PAY  POINT BIRTH    CHECK
      1 000001 윤 택 중    850000  80.00 10/10/37 .T.
      2 000002 이 흥 룡    420000  65.50 07/08/61 .F.
      3 000003 박 준 영    340000  74.00 08/06/63 .T.
      4 000004 채 수 태    640000  85.50 12/27/58 .T.
      5 000005 김 규 봉    430000  80.00 12/18/62 .F.
      6 000006 최 영 숙    436000  74.00 05/13/66 .F.
      7 000007 김 현 진    614000  65.50 04/19/65 .T.
      8 000008 송 진 호    366500  65.50 07/31/60 .T.
      9 000009 최 현 호    366800  80.00 02/01/61 .F.
     10 000010 김 상 넝    807000  65.50 02/23/63 .T.
     11 000011 김 용 석   1240000  74.00 09/25/58 .F.
     12 000012 장 직 순    520000  80.00 06/24/56 .F.
     13 000013 이 정 민    690000  90.00 03/21/55 .F.
     14 000014 전 경 호    570500  90.00 11/20/56 .T.
     15 000015 김 동 균    492500  65.50 10/04/57 .T.

. select 2      && 다시 2 번 작업 영역에 돌아와 데이터를 읽는다.

. list
Record# CODE   TEL        HMAIL
      1 000001 471-3378   tg032
      2 000002 877-5283   sbpark
      3 000003 305-3355   tak
      4 000004 232-1764   grah0
      5 000005 485-8058   kskye
      6 000006 555-4655   love1
      7 000007 757-3284   home74
      8 000008 477-2014   spkm6p
      9 000009 862-5125   hahnkm
     10 000010 688-1994   ahn01dh
     11 000011 682-1122   swj57
```

```
     12  000012 785-4625   leeks
     13  000013 591-6366   choi
     14  000014 832-5919   sambl
     15  000015 566-1861   skuh5p

. select 1        &&----- 1 번 작업영역으로 돌아와서 7 번째 레코드를 읽는다.
. go 7
. display
Record#  CODE   NAME            PAY   POINT BIRTH     CHECK
      7  000007 김 현 진    614000  65.50 04/19/65 .T.

. select 2      &&----- 2 번 작업영역으로 돌아와서 7 번째 레코드를 읽는다.
  go 7
  display
  cord#  CODE   TEL          HMAIL
      7  000007 757-3284     home74

. select 1   &&----- 다른 작업영역의 데이터와 함께 볼 수도 있다.
                   언제라도 작업영역과 필드를 함께 사용하면 다른 작업영역에
                   있는 데이터를 읽어 올 수가 있다. 현재 작업영역2의
                   레코드 위치는 7 번이기 때문에 7번 레코드에 있는 데이터를
                   를 읽어 온다.
                   주의 : 2->tel 처럼 숫자는 사용할 수 없다.
. display code,name,pay,point,birth,b->tel,b->hmail,check
Record#  code   name          pay point birth     b->tel    b->hmail check
      7  000007 김 현 진 614000 65.50 04/19/65 757-3284 home74    .T.

. select 2   &&----- 2번 작업 영역에서 1번 작업영역의 내용과 함께 읽었다.
. display code,a->name,tel,hmail,a->point,a->pay
Record#  code   a->name  tel       hmail  a->point  a->pay
      7  000007 김 현 진 757-3284 home74     65.50  614000

. select 1                                  && ----   1번 작업영역에서 2번
. list code,name,b->tel,b->hmail                      작업영역의 내용과
Record#  code   name      b->tel     b->hmail 함께 전체를 읽으면
      1  000001 윤 택 중  757-3284   home74   1번 작업 영역의 것은
      2  000002 이 흥 룡  757-3284   home74   전체를 읽으나 2번
      3  000003 박 준 영  757-3284   home74   작업의 영역은
      4  000004 채 수 태  757-3284   home74   움직이지 않고 계속
      5  000005 김 규 봉  757-3284   home74   현재 위치해 있는
      6  000006 최 영 숙  757-3284   home74   레코드번호인 7번
      7  000007 김 현 진  757-3284   home74   레코드의 내용만
      8  000008 송 진 호  757-3284   home74   나온다.
      9  000009 최 현 호  757-3284   home74
     10  000010 김 상 녕  757-3284   home74
     11  000011 김 용 석  757-3284   home74
     12  000012 장 직 순  757-3284   home74
     13  000013 이 정 민  757-3284   home74
     14  000014 전 경 호  757-3284   home74
     15  000015 김 동 균  757-3284   home74
```

〔리스트 5〕

```
. select 2
. inde on code to code02
  100% indexed          15 Records indexed
. select 1
. set relation to code into b &&---- 1번 작업 영역이 움직일 때 코드에
                                     의해 2번 작업영역이 따라 움직이게
                                     한다.

. list code,name,b->tel,b->hmail
Record#  code   name       b->tel     b->hmail
      1  000001 윤 택 중   471-3378   tg032
      2  000002 이 흥 룡   877-5283   sbpark
      3  000003 박 준 영   305-3355   tak
      4  000004 채 수 태   232-1764   grah0
      5  000005 김 규 봉   485-8058   kskye
      6  000006 최 영 숙   555-4655   lovel
      7  000007 김 현 진   757-3284   home74
      8  000008 송 진 호   477-2014   spkm6p
      9  000009 최 현 호   862-5125   hahnkm
     10  000010 김 상 녕   688-1994   ahn01dh
     11  000011 김 용 석   682-1122   swj57
     12  000012 장 직 순   785-4625   leeks
     13  000013 이 정 민   591-6366   choi
     14  000014 전 경 호   832-5919   sambl
     15  000015 김 동 균   566-1861   skuh5p

. set relation to     &&----- 화일의 관계설정을 해제 시킨다.
. list code,name,b->tel,b->hmail          &&--- 2번 작업영역의 현재 위치가
Record#  code   name        b->tel      b->hmail 화일의 마지막은 eof 에 위치
      1  000001 윤 택 중                         해 있으므로 아무런 내용도
      2  000002 이 흥 룡                         나오지 않는다.
      3  000003 박 준 영
      4  000004 채 수 태
      5  000005 김 규 봉
      6  000006 최 영 숙
      7  000007 김 현 진
      8  000008 송 진 호
      9  000009 최 현 호
     10  000010 김 상 녕
     11  000011 김 용 석
     12  000012 장 직 순
     13  000013 이 정 민
     14  000014 전 경 호
     15  000015 김 동 균
```

〈주의〉

RECNO()을 사용할 때는 따라 움직이게 될 데이터베이스 화일이 단독으로 열려 있어야 하고, 공통된 필드명에 의해 재배치를 시키고자 할 때는 따라 움직이게 될 종속화일은 그 필드가 키인 인덱스 화일과 함께 열려 있어야 한다.

코드라는 필드에 의해 화일을 연결시키기 위해 다음의 리스트 5와 같이 화일을 만든다.

지금까지 데이터베이스 화일을 이용한 몇가지 응용을 알아 보았다. 디베이스를 연구하면 연구할수록 더욱 더 그 기능에 놀라움을 가지게 될 것이다. 다음호에서는 디베이스를 이용한 출력예를 알아보고 디베이스 프로그래밍을 위한 기초를 알아 보도록 하자.

아울러 아직 필자도 사용은 해보지 못했지만 DBASE Ⅳ에 대한 리뷰 기사를 연구한 것을 토대로 소개할 것을 약속한다.

〈이번호에 나온 명령어와 함수〉

LIST : 데이터베이스의 레코드와 필드를 화면 또는 프린터에 출력시킨다.
DISPLAY : 현재의 데이터베이스에서 레코드와 필드를 출력시킨다.
GO : 레코드 포인트를 곧바로 지정된 위치로 이동시킨다.
APPEND FROM : 다른 화일로부터 데이터베이스에 레코드를 추가한다.
BROWSE : 화면당 최대 17개의 레코드를 표시해주는 화면 편집기능을 호출한다.
SORT ON : 데이터베이스를 순서대로 재배열시킨다.
INDEX ON : 주어진 키에 데이터베이스의 레코드가 재배열된 것처럼 나타낸다.
SET INDEX TO : 지정된 인덱스 화일을 오픈시킨다.
USE〈database file〉 INDEX〈index file〉: 모든 운용에 사용될 데이터베이스 화일을 지정한다.
SET ORDER TO : 임의의 화일을 폐쇄하거나 다시 열지 않고도 제어화일로 이용하게 한다.
REINDEX : 현재 사용중인 인덱스 화일을 재구성한다.
SEEK : 인덱스 화일에서 지정된 표현에 맞는 최초의 레코드를 찾아 레코드 포인터를 이동시킨다.
FIND : 인덱스된 화일에서 지정한 문자열에 맞는 최초의 레코드 위치에 포인터를 위치시킨다.
CLOSE ALL : 지정된 화일을 폐쇄한다.
CLEAR ALL : 모든 데이터베이스 화일, 인덱스 화일, 포맷 화일과 관련 사항을 닫는다
SELECT : 지정한 작업 영역을 작동시킨다.
SET RELATION TO : 키에 의해 두개의 데이터베이스를 연결시킨다.
ASC(C) : 문자열의 가장 오른쪽 문자의 ASCII 코드값을 표시한다.
UPPER(C) : 영문 소문자를 대문자로 바꾼다.
DTOC(D) : 날짜표시를 문자상태로 바꾼다.
FOUND() : FIND, SEEK, LOCATE, CONTINUE 명령이 레코드를 발견했을때 논리값(.T.)를 출력한다.

배치화일로 키의 변환을

키보드의 키 값을 정의하는데 특별한 소프트웨어나 복잡한 BASIC 프로그램등이 없어도 단지 버전 2.0이상의 DOS와 배치화일 하나만 있으면 충분하다.

PC의 다재다능한 기능중에는 함수키 뿐만 아니라 일반 표준형 키의 값도 재정의 할 수 있는 기능이 있다. 재정의 작업에서 특히 escape(ASCII문자 27) 코드가 쓰이는 부분은 조금 복잡하다. 이 escape코드는 '라인 지움' 명령으로 받아들이기 때문에 애써 작성한 코드나 텍스트를 무심코 지우는 일이 없도록 주의해서 사용하기 바란다.

그리고 재정의 후에는 원래 DOS에 정의되어 있던 값으로 다시 재할당 시켜야 한다. 나름대로 키값을 정의해서 사용하는 에디터프로그램인 경우도 이같은 키값의 재할당 작업이 필요한데 DOS배치화일을 이용하면 이러한 키값의 할당과 재할당에 수반되는 문제점들을 쉽게 해결할 수 있다.

함수키나 일반 키 정의를 할당하기 전에 ANSI드라이버가 반드시 PC에 상주되어 있어야 한다. 첫째로 DEVICE=ANSI.SYS 문장이 들어 있는 CONFIG.COM 화일이 반드시 부트 디스크에 있어야 한다.

CONFIG.SYS 화일은 지정된 프로그램을 DOS로 인스톨 시켜주는 (이경우는 ANSI 드라이버) 프로그램으로서 2.00 이상의 DOS 디스켓에는 ANSI.SYS라는 화일이 있으므로 이 화일을 부트디스크에 반드시 넣어야 한다. 이 화일을 만들고 싶으면 먼저 A드라이브의 부트디스크로 도스프롬프트를 옮긴후 DOS프롬프트, A>상에 다음과 같이 타자하고 <Enter>키를 누른 다음

```
COPY CON CONFIG.SYS
```

다음 내용을 입력한다.

```
DEVICE = ANSI.SYS
```

<ENTER>키를 누르고 <F6> 또는 <Ctrl-Z>를 입력하면 앞에서 작성한 화일이 저장된다.

이 CONFIG.SYS화일은 PC가 ON됐을때 또는 재부트(Ctrl-Alt-Del)시킬때에 자동적으로 읽혀지는 화일이다. 그리고 이 CONFIG.SYS가 들어 있는 디스크는 다른 프로그램이 실행되기 전에 A드라이브에 놓여져 있어야 한다.

EDLIN과 같은 에디터를 이용해서 배치프로그램 KEYS.BAT(리스트 1 참조)를 작성하고 A드라이브에 있는 디스크에 저장한다. DOS 프롬프트상에서 KEYS ON 또는 Keys on을 입력하면 배치화일이 실행되면서 우리가 지정한 대로 키값이 정의된다. 그리고 KEYS OFF 또는 Keys off라고 입력하면 원래의 DOS에서 정의된 키값으로 재할당 된다.

KEYS.BAT 내용

KEYS.BAT의 맨 처음 라인은 주석문을 나타내는데, DOS는 마침표로 시작되는 문장을 코멘트문(주석)으로 처리한다. 두번째 라인에서의 ECHO OFF는 DOS가 배치화일을 실행할 때 처리되는 명령어가 화면에 디스플레이되지 않음을 나타낸다. 다음 4 라인에 걸쳐 '%1'이 나타나는데 이 표시는 화일이름 다음에 들어가는 파라미터를 의미한다. 예를 들어 Keys on이 입력 되었다면 KEYS.BAT이 로드 된 다음 %1이 on이란 값을 취하는 것이다. 이 문장은 %1이 ON, on, OFF, off중 하나를 선택케 해서 그 선택에 해당하는 테이블, 즉 : ON, : OFF, : END중의 한 곳으로 분기하도록 한다. (만일 아무런 값이 주어지지 않은 경우는 %1과 같게 된다.) 다음에 ECHO가 다시 on으로 세트되는데 다음에 따라오는 PROMPT DOS 명령어를 통해 키할당 작업을 완수하려면 ECHO를 ON으로 설정해야 한다.

키 변환

PROMPT란 명령은 종종 표준형 DOS프롬프트를 편리하게 조작하는데 이용된다. 예를 들어 PROMPT=WHAT NEXT?라고 하면 DOS는 프롬프트를 A> 대신 WHAT NEXT?로 바꾼다. KEYS.BAT에서 PROMPT명령어는 $e 기호를 이용함으로써 escape 코드(ASCII문자 27)를 활성화시키는데 일반적으로 키를 정의하는데 ANSI.SYS가 쓰일때 이 코드가 꼭 필요하다.

키들은 배치화일의 : ON 레이블로 시작되는 처음 부분에서 정의되고 있다. 만일 리턴기능을 자동적으로 얻고 싶으면 키정의 다음에 : 13P를 덧붙이고 그렇지 않은 경우는 P만 쓴다.

거의 대부분의 키의 값이 재정의 가능하며 키의 값을 변형하려면 그 키를 나타내는 ASCII코드를 특별한 escape순서에 의해 ANSI 드라이버로 보내야 한다.

A 또는 백슬래시 같은 표준 키의 ASCII 코드는 오직 1 캐릭터 길이로 그 정의 순서는 다음과 같다.

〔리스트 1〕 KEYS.BAT

```
KEYBOARD KEY REASSIGNMENT BATCH PROGRAM BY PC

echo off
if %1 == ON goto ON
if %i == on goto ON
if %1 == OFF goto OFF
if %1 == off goto OFF
GOTO END
:ON
echo on
prompt=$e[0;59;"DIR "p
prompt=$e[0;60;"TYPE "p
prompt=$e[0;61;"COPY "p
prompt=$e[0;62;"ERASE "p
prompt=$e[0;63;"A: "p
prompt=$e[0;64;0;64p
prompt=$e[0;65;"B: "p
prompt=$e[0;66;"C: "p
prompt=$e[0;67;"D: "p
prompt=$e[0;68;"A:GWBASIC ";13p
prompt=$e[0;120;"A:KEYS ON";13p
prompt=$e[0;129;"A:KEYS OFF";13p
.THE FUNCTION KEYS HAVE BEEN DEFINED
goto END
:OFF
echo on
prompt=$e[0;59;0;59p
prompt=$e[0;60;0;60p
prompt=$e[0;61;0;61p
prompt=$e[0;62;0;62p
prompt=$e[0;63;0;63p
prompt=$e[0;64;0;64p
prompt=$e[0;65;0;65p
prompt=$e[0;66;0;66p
prompt=$e[0;67;0;67p
prompt=$e[0;68;0;68p
.THE FUNCTION KEYS HAVE BEEN DISABLED
:END
prompt $n$g
```

```
prompt = $e[50:"정의"p
```

또한 PC는 키보드에 있는 특별한 키를 나타내기 위해 확장된 ASCII 코드를 취할 수 있다. 이들 특별한 키들(함수키나 또는 shift키와 함께 쓰는 Alt, Ctrl등)은 두 캐릭터 길이를 가지며 0(제로)를 선두로 취한다. 그 정의 순서를 보면 다음과 같다.

```
prompt = $e[0;59"DIR "p
```

ANSI 드라이버

그러면 어떻게 해서 원래의 DOS정의로 변환하고 있을까? 이 과정을 이해하려면 먼저 ANSI드라이버의 역할을 알아야 한다.

키보드에서 A란 키를 눌렀을 때 그 A란 글자가 컴퓨터 속으로 보내지는 것은 아니다. 즉 A문자에 해당하는 ASCII코드가 보내지는 것이 아니라 그 키의 위치에 의해 결정되는 스캔코드로써 보내진다. 그러면 키보드와 PC의 중간 부분에서 ANSI드라이버는 이 스캔코드를 재할당된 리스트와 비교한다. 만약 그 키가 재할당되어 있지 않다면 ANSI드라이버는 그 키에 해당하는 원래 ASCII코드를 PC로 보낸다. 그렇지만 그 키가 재할당되어 있으면 ANSI드라이버는 그 키에 주어진 대로 캐릭터 스트링을 보내게 된다.

키의 값을 원래 DOS에서 정의 해 준 값으로 환원시키는 문제는 최종 정의된 내용을 중복함으로써 해결된다. 예를 들어 리스트 1에서 : ON레이블로 시작되는 첫번째 prompt명령(prompt=$e[0 ; 59 ; "DIR"P)은 ANSI드라이버에게 해당 ASCII코드를 만나면 대신 DIR이란 문자를 보내도록 하는 명령이다. 마찬가지로 : OFF레이블로 시작되는 첫번째 prompt 명령(prompt=$e[0 ; 59 ; 0 ; 59P)에 의해 ANSI 드라이버는 해당 코드를 0 ; 59로 보낸다. 컴퓨터는 ASCII 코드가 중간에서 그 키에 해당하는 DOS정의(0 ; 59)로 변환하는데에 전혀 무관하게 된다. 그리고 키를 원래 정의된 값으로 재할당하는 작업에서는 자동리턴은 포함되지 않는다.

Off와 On

리스트 1에서 <Alt>–1 조합코드(ASCII코드 0 ; 120)는 A : KEYS ON을, 그리고 <Alt>–0 조합코드(ASCII 코드 0 ; 129)는 A : KEYS OFF를 설정한다. 이 ASCII코드들은 : OFF레이블로 시작되는 부분에 나타나 있지 않기 때문에 KEYS OFF 명령을 써도 변하지 않게 된다. 일단 KEYS ON이 실행되면, <Alt>–0를 키보드로부터 입력한다 해도 <Alt>–1으로 변환되어 DOS정의로 보내진다.

이렇게 키의 값을 새로 정의하고 다시 원래의 DOS에서 갖는 값으로 재정의 작업이 끝나면 KEYS.BAT는 프롬프트를 원래의 A>로 되돌리는 : END레이블로 분기한다. 그러나 <Ctrl>–<Break>키로 실행을 중단시키면 프롬프트는 화면에 나타나지 않는다.
이런 경우는

```
prompt = $p$g
```

를 입력하면 다시 화면에 프롬프트가 나온다.

키의 정의는 임의대로 할 수 있으나 한가지 제한이 있다면 ANSI 드라이버 정의에 있어서 190자를 초과할 수 없다는 점이다. 만일 190자가 넘으면 디스크속의 COMMAND.COM에 마치 겹쳐 써지는 것과 같아서 컴퓨터가 종종 마비되어 버리는 경우가 있다. 이 경우는 할 수 없이 스위치를 끄고 다시 ON시키는 수 밖에 없다.

가끔 PROMPT명령 이용시 'Out of environment space'라는 에러메시지가 나올 때가 있는데 이 경우는 컴퓨터가 마비되는 것이 아니라 단지 키의 정의가 실패로 돌아갔음을 알리는 것 뿐이다. 이 에러메시지에서 보이는 엔바이런먼트(environment)란 오퍼레이팅

〔리스트 2〕 AUTOEXEC.BAT

```
echo off
date
time
echo on
prompt=$e[0;59;"DIR "p
prompt=$e[0;60;"TYPE "p
prompt=$e[0;61;"COPY "p
prompt=$e[0;62;"ERASE "p
prompt=$e[0;63;"A: "p
prompt=$e[0;64;0;64p
prompt=$e[0;65;"B: "p
prompt=$e[0;66;"C: "p
prompt=$e[0;67;"D: "p
prompt=$e[0;68;"A:GWBASIC ";13p
prompt=$e[0;120;"A:KEYS ON";13p
prompt=$e[0;129;"A:KEYS OFF";13p
.THE FUNCTION KEYS HAVE BEEN DEFINED
prompt $n$g
echo off
mode 80
verify on
graphics
cls
ver
vol a:
```

시스템이 정보를 기억하는 장소를 말하며 이 정보들은 다른 프로그램이 액세스할 수 있게 해주는 곳이다.(이것은 위에서 설명했던 ANSI 정의 영역과는 다른 것임)

DOS의 SET 명령어로 이 영역을 디스플레이 시킬 수 있는데 예를 들면

```
COMSPEC = A:\COMMAND.COM
PATH = A:B:C

prompt = $n$g
```

과 같은 것이다.

PC가 작동되면 이 엔바이런먼트 영역으로 127바이트를 할당한다. 만일 더 많은 영역이 필요할 때는 DOS는 더 많은 영역을 확보하려고 한다. 그러나 MODE, PRINT 또는 GRAPHICS와 같은 램상주 프로그램이 실행되면 이 영역은 사용할 수 없게 되고 Out of environment란 에러메시지가 디스플레이 된다. 본고에서 소개한 KEYS. BAT인 경우는 RAM상주 프로그램이 실행된 후에 너무 많은 PROMPT 명령문이 엔바이런먼트 영역에 채워질 때 이 에러 메시지를 출력한다.

이같은 문제를 해결하는 방법은 두가지가 있다. 한가지는 RAM 상주 프로그램을 실행하기에 앞서 AUTOEXEC.BAT화일 내에 키정의 부분을 포함시키는 것이다(리스트 2 참조). 또 한가지 방법은 비록 복잡하기는 하나 멋있는 방법이라고 볼 수 있다. escape코드(ASCII 문자 27)를 쓸 수 있는 에디터를 가지고 있다면 에디터 내로 KEYS.BAT를 읽어 들인 후 :END 레이블로 시작되는 부분의 prompt=ng를 모두 제거하고 이 부분을 에디터의 escape 코드와 ECHO로 교체한다. 이 ECHO 명령은 키정의 내용을 출력해준다. 이와 같이 prompt 명령을 변형시켜 사용하면 엔바이런먼트 영역도 같이 변형된다. PC

노턴 코맨더

양왕성 /성균관대학교 수학과

우리 주변에는 수많은 유틸리티들이 쏟아져 나오고 있다.

이들 유틸리티들이 어떤 기능들을 갖고 있는가 한번씩 짚고 넘어간다면 컴퓨터와의 만남이 더욱 재미있을 것이다. 또한 유틸리티들은 독자들이 프로그래밍 작업을 하는데에도 큰 도움을 줄 수 있다. 주위의 컴퓨터 사용자들이 즐겨쓰는 프로그램이 있다면 어떤 이유에서 그것을 고집하고 있는지 그 이유를 물어 보기 바란다. 비록 그 프로그램을 직접 쓰고 있지는 않더라도 그 이유를 분석해 보면 좋은 경험이 될 것이다.

여기서 소개하려는 노턴 코맨더(이하 NC라 칭함)는 노턴 유틸리티를 만든 피터 노턴(Piter Norton)에 의해 만들어진 재미있는 프로그램이다. NC는 램상주 프로그램으로 DOS 명령을 그대로 쓸 수 있으면서 화일관리를 할 수 있게 해 주는 특이한 프로그램 환경을 제공한다. 단지 몇개의 키조작으로 COPY, RENAME, DELETE, TYPE 뿐만 아니라 편집기능까지 갖추고 있으며 자주 쓰이는 명령어를 메뉴로 만들어서 실행할 수 있고 마우스까지 지원해 컴퓨터 사용자들과 쉽게 친해질 수 있으리라 생각된다.

NC디스크에는 다음과 같은 내용의 화일들이 들어있다.

NC.EXE: 주 실행 화일
NC.EXT: 확장자 실행화일
NC.INI: 패널의 형태를 설정하는 화일
NC.MNU: 유저가 정의하는 메뉴화일
NCSMALL.EXE: 메모리의 효율적인 활용을 위한 로더 화일

하드디스크를 갖고 있는 독자는 하드디스크에 이 화일들을 설치한다. 기존의 노턴 유틸리티 디렉토리가 있으면 이 디렉토리(예 : C : \NU)를 이용하는 것도 좋은 방법이다.

NC의 실행

DOS프롬프트 상에서 다음과 같이 입력한다.

```
A:\>NC
```

NC가 실행되면서 화면의 상단에 두개의 윈도우가 나타나고 그 밑에 DOS 코맨드 라인이 그리고 화면의 맨 밑 줄에는 함수키들이 디스플레이 된다 (화면 1 참조).

이들 윈도우를 패널이라 부르는데 좌측은 처음 NC를 부팅시킨 드라이브 (여기서는 A:\)의 디렉토리 패널을 나타내고 우측에는 상태표시 패널이 표시된다. 이들 패널 형태는 나중에 설명할 함수키(F9)에 의해 사용자가 임의로 선택할 수가 있다.

디렉토리 패널에서 대문자로 나타낸 것은 서브디렉토리를 의미하며 소문자로 나타낸 것은 화일을 의미한다. 그리고 ▦는 숨겨진 화일(Hidden File)을 나타내고 (..)은 상위 디렉토리를 나타낸다.

상태표시 패널은 다시 세부분으로 분리되어 있으며 첫번째 윈도우에는 노턴 코맨더의 버전과 Copyright정보가 들어 있고 두번째 윈도우에는 디스크 및 시스템 메모리에 대한 정보가 들어가게 된다. 끝으로 세번째 윈도우는 디렉토리를 기억하기 쉽게 작성해 두는 레이블과는 같은 것이다.

만약 화면에 상태표시 패널이 나온 상태라면 편집 키인 〈F4〉를 누르면 직접 이 윈도우 속으로 들어가 편집을 할 수 있게 된다.

〈ENTER〉키를 이용한 파라미터 전달

키보드에서 좌측 키패드의 키들은 패널을 조작할 때 사용되고 일반 키들은 DOS 프롬프트 라인에서 DOS 명령을 내릴 때 사용된다.

선택 막대로 디렉토리를 지정하고 〈ENTER〉키를 누르면 해당 디렉토리로 이동한다. 즉 DOS의 CD(Change Directory) 명령과 같다. 마찬가지로 (..)을 선택하여 〈ENTER〉를 누르면 DOS의 CD..과 같은 역할을 수행한다. 맨 위의 루트디렉토리로 이동하는 CD\와 같은 기능은 〈CTRL〉–\이다.

화일의 확장명이 BAT, COM, EXE일 때는 선택 막대로 화일을 선택하고 〈ENTER〉를 누르면 즉시 실행한다. 그럼 패널 내의 화일이름을 프롬프트 라인으로 복사해 보자. 이때는 〈CTRL〉–〈ENTER〉를 입력한다. 예를 들어 에디터에 있는 화일이름을 파라미터로 전달할 경우 프롬프트 라인에서

```
C:\>TYPE
```

라고 입력한 상태에서 선택 바를 움직여 원하는 화일(예 : AUTOEXEC.BAT)을 선택하고 〈CTRL〉-〈ENTER〉를 누르면

```
C:\>TYPE AUTOEXEC.BAT
```

와 같이 된다. 이 상태에서는 〈ENTER〉를 누르기만 하면 실행한다. 그밖의 〈CTRL〉키 사용방법은 표 1을 참조하기 바란다.

함수키 사용 방법

실질적인 NC의 이용은 화면 밑에 디스플레이된 10개의 함수키로 이루어진다. 그러므로 NC를 효율적으로 이용하기 위해서는 이들 함수키들을 능숙하게 처리할 수 있어야 한다.

● **F10(Quit)**

순서가 좀 이상하지만 〈F10〉부터 알아 보자. NC를 처음 대하면 넓은 화면을 쓰던 독자들은 조금 답답할 것이다. 또는 다른 프로그램과 상충되어 시스템이 마비되어 버릴 경우에 처하면 언제든지 〈F10〉를 눌러 NC 프로그램을 RAM으로부터 제거시킬 수 있다.

대부분의 램상주 프로그램이 그렇듯이 프로그램을 램에서 제거할 때는 RAM에 설치하는 역순으로 하는 것이 안전하다. 그리고 RAM에서 메모리를 활용하는 프로그램을 맨 나중에 설치할 것을 권한다. 안전하게 제거되는 것을 확인했으면 다시 NC를 올려 연습해 보자.

● **F1(Help)**

진행방법이 기억나지 않거나 키가 생각나지 않으면 〈F1〉을 누른다. 각 상황에 따라서 도움말이 나온다(화면 2 참조). 도움말 화면이 나온 후 아무키나 누르면 원래 화면으로 돌아간다.

● **F2(Menu)**

사용자 정의 메뉴(User Define Menu)이다. 〈F2〉를 누르면 NC.MNU화일을 찾는데 이 화일이 없으면 실행할 수 없다. 현재 디렉토리에 NC.MNU가 없으면 루트에서 NC.MNU를 찾는다. 이것이 단점이 될 수도 있으나 도리어 각 디렉토리마다 필요한 메뉴를 바꿔 넣을 수 있다는 장점이 될 수도 있다.

일단 NC.MNU에는 2개의 샘플 메뉴가 있는데 〈F2〉를 눌러 보자(화면 3 참조). 이때의 메뉴는 리스트 1에 의해 자동으로 만들어진 실행가능한 메뉴로서 좌측의 문자를 지정하든지 메뉴 막대로 선택해서 실행시킬 수도 있다. 취소는 〈ESC〉키를 누르면 된다.

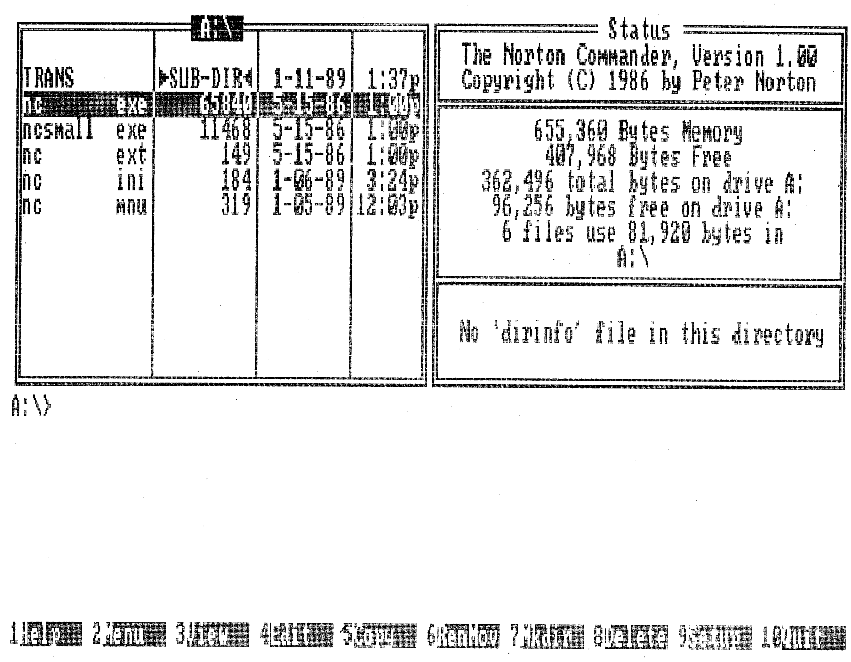
콜론(:)의 왼쪽에는 하나의 키(여기서는 D나 F1)로 실행가능한 키를 지정하며 오른쪽에는 디스플레이되는 문장을 작성한다. 이것은 곧 메뉴 막대로 움직여서 선택되는 문장이다. 다음 줄에는 DOS상에서 실행될 수 있는 명령어를 서술해 주며 여러개의 명령어를 지정할 수 있고 다음 순서의 메뉴와 구분하기 위해 한줄을 비워둔다. 이와 같은 방법으로 20개까지 메뉴를 만들 수 있다.

리스트 2는 NC.MNU를 수정한 것으로 원래의 내용과 비교하면서 독자들 스스로가 각 디렉토리에 맞춰 작성해 보기 바란다.

● **F3(View)**

DOS상에서 한 페이지가 넘는 리스트

〔화면 1〕 NC의 실행화면



〔표 1〕 〈CTRL〉키 이용방법

입력하는 키	내용
^I	반대편 패널로 선택 막대를 옮김
^U	왼쪽과 오른쪽 패널 위치를 서로 바꾼다.
^P	선택 막대가 없는 쪽 패널을 안보이게 한다.
^L	선택 막대가 없는 쪽 패널을 상태표시 패널로 바꾼다.
^O	패널 전체를 안보이게 한다.
^B	화면 밑에 보이는 함수키들을 안보이게 한다.

(＊모두 토글방식으로 동작함) (표시는 〈CTRL〉키를 의미)

를 TYPE 명령으로 보기 위해서는 〈CTRL〉-〈S〉를 누르면서 확인하고 지나가면 다시 TYPE를 반복해야 하는 번거로움이 있다. 물론 전문 에디터를 사용하면 되지만 그 크기와 속도 때문에 TYPE 명령을 대용하기에는 절차가 복잡하다. 이때 〈F3〉을 누르면 만족할 만큼의 속도로 원하는 리스트를 확인할 수 있다. 선택 바를 움직여 화일을 선택하고 〈F3〉을 누르면 화일을 담고 온다.

〈Shift〉-〈F3〉은 화일이름을 먼저 입력받고 View 모드로 들어 간다. 이 경우는 화일들이 섞여있어 찾기 힘들 때 사용하면 편리하다.

참고로 View 모드에서의 키 사용방법을 표 2에 나타내었다.

● **F4(Edit)**

편집 모드로서 위의 〈F3〉과 같은 방법으로 화일을 선택하고 〈F4〉를 누르면 된다. (또는 〈Shift〉-〈F4〉) 이것은 간이 에디터이며 편집 모드로 들어가는데 속도만큼은 최상급이라 하겠다. 왜냐하면 에디터기능은 이미 NC에 올라와 있고 화일정보도 NC가 가지고 있어 단지 화일내용만 로드시키면 되기 때문이다. 예를 들어 배치화일의 내용을 수정하고 싶으면 선택막대로 AUTOEXEC.BAT를 선택하고 〈F4〉키를 입력해 보라. 순식간에 편집모드로 변환되었을 것이다.

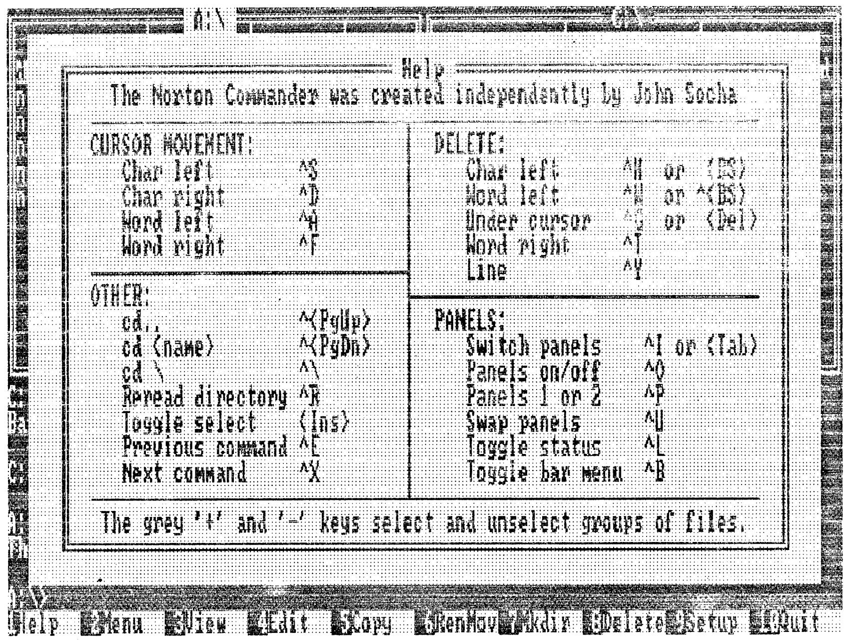
앞에서도 언급했듯이 편집은 단순명령으로 수행되므로 일부 부분을 수정할 때 유용하다. 커서를 옮기는 명령과 지우는 명령은 대부분의 에디터와 같다. 특이한 것은 〈CTRL〉-〈K〉의 기능이 라인을 지우는 명령으로 쓰이고, 언제나 〈Insert〉 모드상태라는 점이다. 표 3은 편집 모드에서의 주요 키 기능을 나타낸 것이다.

● **F5(Copy)**

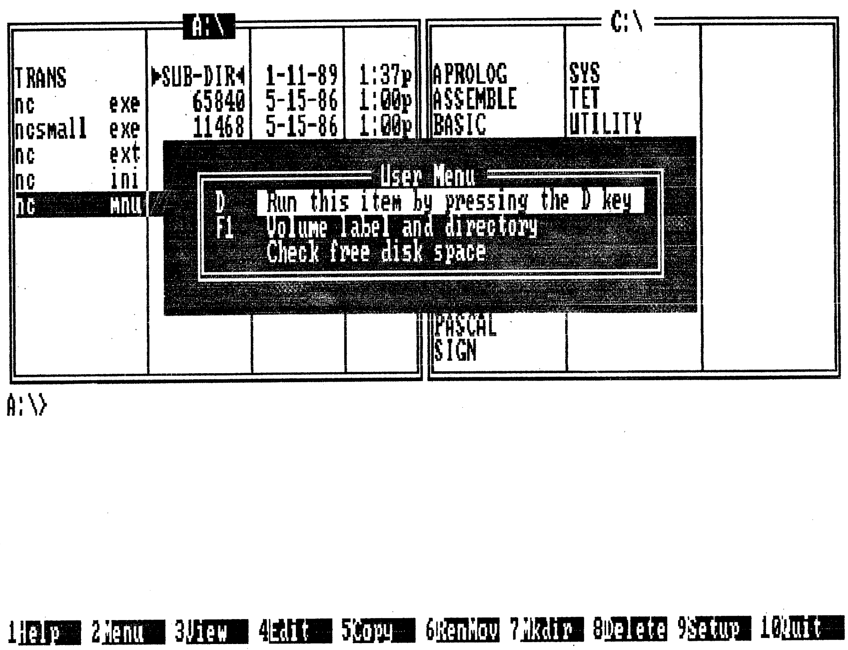
① 하나의 화일을 복사하는 경우

복사할 화일을 선택한 다음 〈F5〉를 또는 〈Shift〉-〈F5〉를 누르면 기본적으로 반대쪽 패널의 디렉토리가 복사

〔화면 2〕 도움말 화면



〔화면 3〕 NC.MNU의 화면



〔리스트 1〕 NC.MNU의 리스트 예

```
D:   Run this item by pressing the D key        dir
      dir/w
F1:  Volume label and directory                 Check free disk space
      Vol                                            chkdsk
```

〔리스트 2〕 NC.MNU의 수정

```
D:      Run this item by pressing the D key
        dir /w

F1:     Volume label and directory
        Vol
        dir

V:      Dos Version
        ver

P:      Path name
        path

Check free disk space
        chkdsk

' This is a comment. Any line that starts with a single quote
' is a comment line, and is ignored by the Commander.
```

〔표 2〕 View모드에서의 키 사용

커서 이동 키	제어키	동 작
↓	^X	한줄 밑으로 이동
↑	^E	한줄 위로 이동
→	^D	1칼럼 우측으로 이동
←	^S	1칼럼 좌측으로 이동
^→	^F	40칼럼 우측으로 이동
^←	^A	40칼럼 좌측으로 이동
Home	^Home	같은 칼럼에서 첫 라인으로 이동
End	^End	같은 칼럼에서 마지막 라인으로 이동
^PgUp	^R	전 페이지로 이동
^PgDn	^C	다음 페이지로 이동
F7		스트링 탐색
Shift-F7		탐색을 계속 유지한다.

될 목적지가 된다.(목적지를 DOS프롬프트 라인에서 직접 입력해도 된다.) 〈ENTER〉키를 치면 실행되는데 DOS의 COPY명령과 같은 기능을 갖는다.

② 여러 화일을 복사하는 경우

DOS상에서 와일드카드 문자(*나?)를 이용해서 여러 화일을 선택하듯 NC에서도 여러화일을 저정할 수 있다. 즉 키패드의 키중에서 플러스(+:Gray Plus)키를 눌러 화일 형식을 적어 주면 그 화일 형식에 맞는 화일들이 선택된다. 선택된 화일들도 약간 밝은색으로 되고 만약 선택막대가 특정 화일 위에 있다면 막대는 밑줄(Under Line)로 변한다. 반대로 와일드카드 문자로 화일선택을 취소할때는 키패드의 마이너스(-:Gray Minus)를 누르고 화일 형태를 지정해 주면 그 형식에 맞는 화일들은 선택이 취소된다. 그리고 화일 하나하나를 선택 또는 취소할 때는 〈Ins〉키를 누른다. 〈Ins〉키는 화일 선택을 토글방식 즉 선택된 것은 취소시키고 선택되지 않은 선택이 되게 한다. 〈Ins〉, [+], [-]키 사용방법은 다음 함수키(F6, F8)에도 적용이 된다. 화일선택이 끝나면 〈F5〉를 눌러 목적지를 정하고, 〈ENTER〉를 누른다. 물론 중복되는 화일인 경우는 경고메시지를 출력한다.

● **F6(Rename or Move)**

화일 선택방법은 〈F5〉와 같고 기능은 DOS의 REN(Rename)명령 또는 MOVE 와 같다.

〈F5〉를 누르면 반대쪽 패널의 디렉토리가 기본으로 나타나는데 여기에 디렉토리를 쓰면 Move(즉 카피 그리고 삭제)가 되고 화일이름을 지정하면 Rename 명령이 된다. 화일을 정리할 때 중복되지 않도록 다른 디렉토리로 복사를 한 후 원래의 디렉토리의 화일을 지워버리는 것이 move 기능이다. 〈Shift-F6〉은 화일을 직접 지정한다. 그리고 DOS의 RENAME 명령과 틀린점은 화일이름 대신 디렉토리이름을 지정하면 디렉토리 이름까지 바꿀 수 있다는 것이다. 그러므로 디렉토리이름을 지정하면 디렉토리 이름까지 바꿀 수 있다는 것이다.

● **F7(Create Directory)**

현재의 디렉토리에서 서브디렉토리를 만들어 준다. DOS의 MD 명령, Mkdir 명령과 같다. 〈F7〉을 누른 후 새로 만들 디렉토리의 이름을 적어주면 된다.

● **F8(Delete)**

DOS의 Del 명령과 같다. 화일 선택은 (F5)와 같이 하나 또는 여러개의 화일을 지정할 수도 있다. 여러개의 화일을 지정했을 때는 주의해서 사용해야 한다. 디렉토리를 지정하면 RD(Remove Directory) 명령과 같아진다. 물론 그 디렉토리는 비어 있어야 한다. 〈Shift-F8〉은 디렉토리명을 직접 지정할 수 있게 해 준다.

● **F9(Setup)**

NC 화면 즉 패널의 형태를 지정한다. 〈F9〉를 누르면 화면 4와 같고 현재 정의된 것은 밝은색으로 나타난다. 각 패널좌우를 따로 정의할 수 있다. 각각의 패널은 세가지 형태를 정의한다. 그리

〔화면 4〕 패널형태 설정

```
A:\TRANS                    C:\UTILITY
                  Setup
 Left panel visible:     Yes  No
 Panel type:                 Brief  Status
 Sort files by:          Name          Time  Size
 Disk drive:               C  D
 \TRANS
 ------------------------------------------------
 Right panel visible:          No
 Panel type:             Full         Status
 Sort files by:          Name          Time  Size
 Disk drive:             A    D
 \UTILITY
 ------------------------------------------------
 Screen colors:                Color
 Automatic user menu:    Yes
 Prompt:                 A>
 Function key bar:             Off
 ------------------------------------------------
 Press Shift-F9 to save this setup in the NC.INI startup file.

1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9Setup 10Quit
```

〔표 3〕 편집모드에서의 키 사용

동 작	커서 키	콘트롤 키
왼쪽으로 이동	←	^S
오른쪽으로 이동	→	^D
왼쪽으로 한 워드 이동	^←	^A
오른쪽으로 한 워드 이동	^→	^F
1라인 위로 이동	↑	^E
1라인 아래로 이동	↓	^X
1페이지 위로 이동	PgUp	^R
1페이지 아래로 이동	PgDn	^C
첫 라인으로 이동	Home	
마지막 라인으로 이동	End	
화일의 처음부분으로 이동	^Home	
화일의 끝부분으로 이동	^End	

고 그 패널이 보여질 것인지 아닌지, 그리고 화일순서를 어느 것을 기준으로 할 것인가를 지정한다. 사용할 드라이브를 선택할 수도 있는데, 프롬프트 라인에서 바로 드라이브명을 써도 된다. 그외에 화면 형태라든가 프롬프트 형태, 함수키들의 출력 유무 등을 정한다. 〈Shift–F9〉은 패널 형태를 저장한다. 관련 화일은 NC.INI이다. 만일 보관을 하지 않은 채 다시 NC를 작동시키면 원래의 화면이 나타난다.

NC.EXT

화일을 선택하고 〈ENTER〉를 눌렀을 때 특정한 확장자명을 만나면 그 확장자에 맞는 프로그램을 자동 실행시킨다.

〈보기〉 NC.EXT의 예

```
prg: pe2 !.!
txt: type !
bas: \GWBASIC\gWbasic !
*: rem no run file this extension
```

NC.EXT의 내용이 리스트 3과 같을 때 선택 막대로 직접 TEST.BAS를 선택하고 〈ENTER〉를 누른다. 그러면 프롬프트라인에 다음과 같이 GWBASIC이 나타나고 TEST.BAS가 실행될 것이다.

```
C:\>\GWBASIC\gwbasic test
```

다시 test.prg를 선택하고 〈ENTER〉를 누르면 다음과 같이 실행된다.

```
C:\>pe2 test.prg
```

즉, 어떤 특정한 확장자를 만났을 때 실행해야할 화일을 지정해 둘 수 있다. 그러므로 어떠한 확장자를 만났을 경우 화일을 파라미터로 전달하는 과정을 정의해두면, 단지 화일을 선택하는 것 만으로도 그 화일을 작업할 수 있다. 형식은 먼저 확장자를 쓰고 콜론(:) 한다음 실행해야할 명령과 선택된 화일형식을 맞춰 적으면 된다.

(화일형식)

- ! – 확장자 없는 화일이름
- !.! – 확장자 있는 화일이름
- !: – 현재의 드라이브명
- ! – 현재의 PATH 이름

(예)

```
C:\>NC test.ext
test
test.ext
C:
NC
```

그리고 지정된 확장자가 아닌 경우는 다음과 같이 마지막줄의 메시지가 출력된다.

```
rem no run file this extension
```

NC.EXT의 LIST 3과 LIST 4를 비교해 보고 독자들이 직접 NC.EXT를 디렉토리에 맞춰 수정해 보기 바란다. NE.EXT는 NC가 실행될 때 체크되므로 NC.EXT의 수정된 것을 사용하려면 NC를 다시 실행시켜야 한다.

NC는 RAM 상주 프로그램으로 상당한 메모리 용량을 차지하는데 이런 이유로 메모리가 모자라 실행하지 못

〔리스트 3〕 NC.EXT의 내용(1)

```
prg: pe2 !.!
txt: type !
bas: \GWBASIC\gwbasic !
*:   rem no run this file
```

〔리스트 4〕 NC.EXT의 내용(2)

```
doc: word !
bak: word !.!
wks: \ver_1a\123
wk?: \ver_2_0\123
xtk: xtalk !
:    wp
*:   rem No run this file
```

하는 프로그램이 있다. 그렇다고 프로그램을 실행할 때마다 NC를 빠져나간 후 다시 실행시킬 수도 없는 일이다. 이때는 NC대신에 NCSMALL.EXE를 실행시킨다. NCSMALL은 NC의 축소된 형태가 아니라 필요로 할 때만 NC를 찾아 메모리에 탑재되는 것이다. 즉 NCSMALL은 NC.EXE의 로더이므로 NC.EXE가 없으면 실행할 수 없다.

결국 NC.EXE를 이용하든 NCSMALL. EXE를 이용하든 기능상의 차이는 없고 단지 응용 프로그램을 실행시킬 경우 NC가 차지했던 메모리를 비워주고 다시 DOS로 돌아오면 자동적으로 NC를 로드하고 실행하는 효율적인 작업 환경을 제공한다는 것이다. 직접NC를 실행시킬 때보다는 속도가 약간 느려지지만 메모리를 효율적으로 이용하는 방법이므로 기억해 두기 바란다.

지금까지 NC를 살펴 보았는데 초보자들에게는 안내자로 실력자들에게는 좋은 도구로 쓰였으면 하는 게 필자의 바람이다. PC

C언어에 있어서의 최적화

장석원/영진출판사 연구원

C언어가 대중적인 프로그래밍 언어가 될 수 있었던 중요한 한가지 이유는 최적화에 강하다는데 있다. 이때의 최적화란 불필요한 코드를 없애고 가장 빨리 실행될 수 있는 구조의 코드를 생성하여 주는 것을 말한다.

인터프리터 베이직이 프로그래밍 언어로서의 대명사를 이루던 예전과는 달리 요 즈음의 프로그래밍 언어는 그야말로 춘추전국 시대를 방불케할 만큼 그 종류가 다양해지고 있다. 코볼이 주종을 이루던 업무용 프로그램 분야도 이제는 데이터 베이스로 대치되고 있고 베이직의 경우도 요즈음의 추세에 맞게 새롭게 단장된 퀵베이직(Quick Basic)이 많은 프로그래머들로부터 애용되고 있는 형편이다.

특히 C 언어의 경우는 어셈블리가 독차지하고 있던 시스템 프로그램 분야를 비롯한 거의 모든 분야에 걸쳐 그 위세를 떨치고 있다. 그 이유는 C 언어 자체의 특징과 날로 강력해지는 C 컴파일러 기능 때문이라고 할 수 있을 것이다. 현재 PC에서 사용되는 대표적인 C 컴파일러로서는 마이크로소프트 C와 터보 C를 들 수 있는데, 그 밖에도 C86 Plus란 컴파일러가 있긴 하지만 우리나라에서는 그리많이 사용되고 있지는 않는 것 같다.

이렇듯 C 언어가 대중적인 프로그래밍 언어가 될 수 있었던 데에는 한가지 중요한 이유가 있다. 그 이유는 C 언어가 그 어떤 언어보다도 최적화(Optimization)에 강하다는데 있다. 따라서 이번호에서는 최적화에 대한 의미를 알아보고 최적화에 따라서 생겨난 어셈블리 리스트 출력 기능을 보다 융통성 있게 사용할 수 있는 프로그램을 소개하고자 한다.

최적화의 의미

컴파일이란 의미가 작성된 프로그램을 컴퓨터가 실행할 수 있는 기계어 코드를 생성하는데 있다는 것은, 컴퓨터를 사용하는 사람이라면 누구라도 알고있는 사실일 것이다. 그런데 예전과 같이 단순히 실행될 수 있는 기계어 코드를 생성해주는 컴파일의 의미와는 달리 요즈음은 얼마만큼 최적화가 잘된 코드를 생성해 주는가하는 또하나의 가치가 부여되어 있다.

이때의 최적화란, 가능한 모든 쓸데없는 코드를 없애므로서 불필요한 코드를 없애고 가장 빨리 실행될 수 있는 구조의 코드를 생성하여 컴파일된 프로그램이 가장 빠르게 실행될 수 있게끔 해주는 것을 말한다. 따라서 이러한 최적화를 얼마만큼 잘해주느냐에 따라 그 컴파일러의 우수성이 좌우된다고 할 수 있을 것이다.

마이크로소프트 C나 터보 C 컴파일러의 옵션중에는 어셈블리 리스트 출력 기능이란 것이 있다. 이 어셈블리 리스트 출력 옵션이란, C 언어로 작성된 프로그램을 어셈블리 소스 형태로 변환해서 출력해주는 것을 말한다. 뿐만 아니라 이때 생성된 어셈블리 소스는 마이크로소프트의 MASM이나 볼랜드의 TASM에 의해서 완벽하게 어셈블될 수도 있다. 다시 말하자면 생성된 어셈블리 소스를 어셈블해서 얻은 오브젝트는 실제 컴파일러에 의해서 얻은 오브젝트와 동일하다는 뜻이 된다.

그렇기 때문에 요즈음의 시스템 프로그래머들 중에서 처리가 복잡한 프로시듀어들은 일단 C 언어로 코딩을 마친뒤 이때 생성된 어셈블리 소스를 참고로해서 다시 보다 최적화된 어셈블리 프로그램을 작성하는 경우도 있다.

어셈블리 리스트의 출력 기능

실제 어셈블리 리스트가 출력된 형태를 보도록 하자. 리스트 1은 입력받은 문자열을 버블소트 알고리즘을 사용하여 소트한뒤, 그 결과를 출력하는 프로그램이다. 그럼 리스트 1을 마이크로소프트 C와 터보 C를 사용하여 출력한 어셈블리 리스트의 형태를 살펴보도록 하자.

1. 마이크로소프트 C를 사용한 경우

마이크로소프트 C의 어셈블리 리스트 출력 옵션은 /Fa 이다. 먼저 리스트 1의 프로그램을 bubble.c란 이름으로 입력한뒤 다음 보기와 같이 /Fa 옵션을 사용하면 어셈블리 리스트를 얻을 수 있을 것이다.

리스트 2는 보기와 같은 옵션을 사용하여 얻은 어셈블리 리스트의 일부를 표시해 놓은 것이다.

리스트 2에서 표시한 것은 실제 출력된 리스트의 일부만

〔리스트 1〕 입력된 분자열을 버블 소트를 사용하여 소트한 뒤 그 결과를 출력하는 프로그램

```
void bubble();
/* sort driver */
main()                /* sorta string from the keyboard */
{
    char s[80];

    printf("Enter a string : ");
    gets(s);
    bubble(s,strlen(s));
    printf("The sorted string is : %s\n",s);
}

void bubble(item, count)      /* bubble sort procedure */
char     *item;
int count;
{
    register  int  a,b;
    register  char t;

    for(a=1; a<count; ++a)
        for(b=count-1; b>=a; --b) {
           if(item[b-1] > item[b]) {
                                 /*  exchange elements */
             t = item[b-1];
             item[b-1] = item[b];
             item[b] = t;
        }
        }
}
```

〈보기〉

C>CL /Fa bubble.asm/c bubble.c 〈cr〉

- C 소스의 이름
- 컴파일만을 하고 링크는 하지 않는다.
- 출력될 화일의 이름
- 어셈블리 리스트의 출력 옵션
- 마이크로소프트 C 컴파일러

을 표시해 놓은 것이다. 완벽한 리스트는 독자 스스로가 충분히 출력해서 얻을 수 있을 것이다. 그리고 리스트 2의 내용중에는 ";Line xx"란 주석이 달려있는 것을 볼 수 있을 것이다. (어셈블리의 주석문은 세미콜론을 사용한다.) 이때 Line xx는 실제 소스의 라인번호를 의미하는 것이고, 따라서 그러한 주석문 이후의 어셈블리 리스트는 주석문에서 지정된 소스의 내용이 어셈블리 리스트로 출력된 것이다.

2. 터보 C의 경우

다음은 터보 C의 예를 보인 것이다.

〔리스트 2〕 마이크로소프트 C를 사용하여 출력한 어셈블리 소스

```
              .
              .
              .
              .
$SG105   DB   'Enter a string : ',  00H
$SG108   DB   'The sorted string is : %s',  0aH,  00H
CONST      ENDS
_TEXT      SEGMENT
    ASSUME    CS: _TEXT
; Line 5
    PUBLIC    _main
_main     PROC NEAR
    push bp
    mov  bp,sp
    mov  ax,80
    call __chkstk
;   s = -80
; Line 8
    mov  ax,OFFSET DGROUP:$SG105
    push ax
    call _printf
    add  sp,2
; Line 9
    lea  ax,WORD PTR [bp-80] ;s
    push ax
    call _gets
    add  sp,2
; Line 10
    lea  ax,WORD PTR [bp-80] ;s
    push ax
    call _strlen
    add  sp,2
    push ax
    lea  ax,WORD PTR [bp-80] ;s
    push ax
    call _bubble
    add  sp,4
              .
              .
              .
```

C>TCC -c -s Bubble.c

- C 소스의 화일이름
- 어셈블리 리스트 출력 옵션
- 컴파일만 하고 링크는 하지 않는다.
- 터보 C 컴파일러

터보 C의 경우도 마찬가지다. 그러나 출력된 어셈블리의 형태는 마이크로소프트 C와는 다를 것이다. 단지 ";Line xx"

식의 주석문만은 동일한 형태로 삽입된다. 그런데 여기서 한가지 불편한 문제가 있다. 그것은 ";Line xx"란 주석문이 나타날때마다 C 소스를 다시 훑어봐야 한다는 것이다. 즉, 두개의 소스 리스트를 번갈아 가면서 봐야 한다는 문제가 발생하는 것이다.

그런데 이러한 문제를 개선하기 위한 좋은 방법이 한가지 있다. 그것은 C 소스와 어셈블리 소스를 머지시키는 것이다. 다시 말하면 ";Line xx"란 주석문대신 실제 C 소스를 대치하는 것이다. 그리고 이러한 번거로운 작업을 프로그래머가 일일이 할 수는 없는 일이므로, 대신 이러한 일을 처리해줄수 있는 프로그램을 소개하도록 하겠다.

머지(MERGE.C) 프로그램

그러면 앞서 말한대로 C 소스와 어셈블리 소스를 머지해주는 프로그램을 소개하도록 하겠다. 리스트 3의 프로그램이 바로 머지 프로그램인데, 컴파일러는 어떤 것을 사용해서 상관없다.

먼저 리스트 3의 프로그램을 입력하여 merge.c라는 이름으로 디스크에 저장한 뒤, 터보 C든 마이크로소트 C든 독자가 사용하기 편리한 컴파일러를 사용해서 컴파일하면 된다. 다음은 각 컴파일러에 따른 컴파일 방법을 간략히 표시해 놓은 것이다.

```
C>TCC merge.c <cr>  ──── 터보 C의 경우
C>CL merge.c <cr>   ──── 마이크로소프트 C의 경우
C>QCL merge.c <cr>  ──── 퀵 C의 경우
```

이번엔 사용법을 알아보자. 사용법은 매우 간단하다. 단지 C 소스의 이름만을 지정해주면 된다. 그런데 여기서 몇가지 지켜야할 사항이 있다. 그것은 지정된 화일 이름에는 절대 확장자명이 붙어서는 안된다는 것이다. 따라서 period(".") 도 붙여서는 안된다.

그리고 반드시 어셈블리 리스트 화일의 이름과 C 소스의 화일이름은 동일해야 하며, C 소스의 확장자명은

〔화면 1〕 bubble.C와 bubble.asm을 머지시키는 과정

```
C:\WP>MERGE bubble

Source code merged successfully!

C:\WP>dir bubble

Volume in drive C is Jang.s.w#01
Directory of  C:\WP

BUBBLE   C      558   8-22-88  12:21a
BUBBLE   ASM   2390   1-04-88   3:23p
BUBBLE   MER   2743   1-04-89   3:40p
    3 File(s)   4720640 bytes free
```

소스의 화일이름 (절대 확장명이 붙어서는 안된다.)

머지 프로그램

bubble.mer이 생성됐다.

〔리스트 3〕 merge.C : C소스와 어셈블리 리스트를 머지시켜주는 프로그램

```
/*  MERGE.C
    Description: merge a 'C' source file with an asm file
*/
#include <stdio.h>
#define   LINEBUFFER 5000          /* length of input line */
#define  NAMESIZE     80           /* DOS file path lenght */
FILE     *p_to_c,
    *p_to_asm,
    *p_to_mer;                  /* line buffer */
char     line_buffer[LINEBUFFER]; /* file name buffer */
char    file_c[NAMESIZE], file_asm[NAMESIZE], file mer[NAMESIZE];
main(argc, argv)
int argc;
char     *argv[];
{
    int  line_count, run_count = 0;
    if   (argc != 2)
         help_me();           /* print the usage messgae */
    strcpy(file_c, argv[1]); /* setup the .C filename   */
    strcat(file_c,".c");
    if(!(p_to_c = fopen(file_c,"r")))  /* open the .C file */
         file_error(file_c);
       strcpy(file_asm, argv[1]);    /* setup the .ASM filename    */
       strcat(file_asm,".asm");
    if(!(p_to_asm = fopen(file_asm,"r")))    /* open the .ASM file */
         file_error(file_asm);
    strcpy(file_mer, argv[1]);    /* setup the .MER filename    */
    strcat(file_mer,".mer");
    if(!(p_to_mer = fopen(file_mer,"w")))    /* open the .MER file */
         file_error(file_mer);
                    /* read each line of the .ASM file */
  while (fgets(line_buffer, LINEBUFFER,p_to_asm))
  {
        if(strncmp(line_buffer, "; Line",6)) /* if "; Line" not found */
```

```
{
     fputs(line_buffer,p_to_mer); /* write it to merge file */
     continue;          /* get next line */
}
line_count = atoi( &line_buffer[7]); /* generate a line count */
if(run_count < line_count)          /* if backlog in .C file */
     for(; run_count < line_count; run_count++)
          write_merge_line(); /* write the lines and catch up */
else
{
     write_merge_line();       /* write corresponding c line */
     run_count++;
          }
   }                        /* close all files */
   fclose(p_to_c);
   fclose(p_to_asm);
   fclose(p_to_mer);
   printf("\n Source code merged successfully! \n");
}
/* write_merge_line()
 * reads a line from the C source, and writes it out preceded by
 * a ";"
 */
write_merge_line()
{
    fgets(line_buffer,LINEBUFFER,p_to_c);   /* get .C file line */
    fputs("; ",p_to_mer);                   /* write "; " to merge file */
    fputs(line_buffer,p_to_mer);            /* put .C line after it */
}
/* help me
   Description: print out the title and usage
*/
help_me()
{
    printf("Merge:\007 a program to merge a 'C' source file and an ASM
file\n");
    printf("Usage: merge file \n");
    printf("where filename is the name of the files to merge\n");
    printf("Do not type in the period or extension.\n");
    exit(0);
}
/* file_error
   Description: prints out error messgae and name of the
                file that could not be opened
 */
 file_error(name)
 char *name;
 {
     printf("Merge:\007 Error in opening file : %s \n",name);
     exit(0);
 }
```

반드시 .C이어야 하고 어셈블리 리스트의 확장자명도 반드시 .ASM이어야 한다.

머지 작업이 이상없이 끝났을 경우에는 확장자명이 .MER인 화일이 생성되는데, 바로 그 화일이 C 소스와 어셈블리 화일이 머지된 것이다.

화면 1은 앞서 예시한 리스트 1의 bubble. c를 머지시킬 경우의 예를 든 것이다. 머지가 완전히 끝났을 경우에는 bubble.mer란 화일을 얻을 수 있을 것 이다(리스트 4 참조).

끝맺는 말

C 언어가 점차 대중적인 언어로서 그 자리를 굳혀가고 있고, 또한 이러한 현상이 바람직하다는게 필자의 견해이다. 그래서 이번호에서는 C 언어를 사용함에 있어서 편리한 몇 가지 기능과 그 기능의 유용성을 알아보았다.

그럼 다음번에 보다 심층적인 내용을 가지고 독자들과 만날 것을 기대하면서 이 원고를 마치도록 하겠다. 끝으로 퍼스널컴퓨터의 창간을 뒤늦게나마 축하드린다. PC

〔리스트 4〕 bubble.mer : 실제 머지된 결과이다. ";Line ××"대신 실제 C소스로 대치된 것을 볼 수 있을 것이다.

```
                 .
                 .
                 .
_TEXT    segment   byte public 'CODE'
; void bubble();
;
; /* sort driver */
; main()      /* sorta string from the keyboard */
; {
_main    proc near
    push bp
    mov  bp,sp
    sub  sp,80
;   char s[80];
;
;   printf("Enter a string : ");
    mov  ax,offset DGROUP:_s@
    push ax
    call near ptr _printf
    pop  cx
;   gets(s);
```

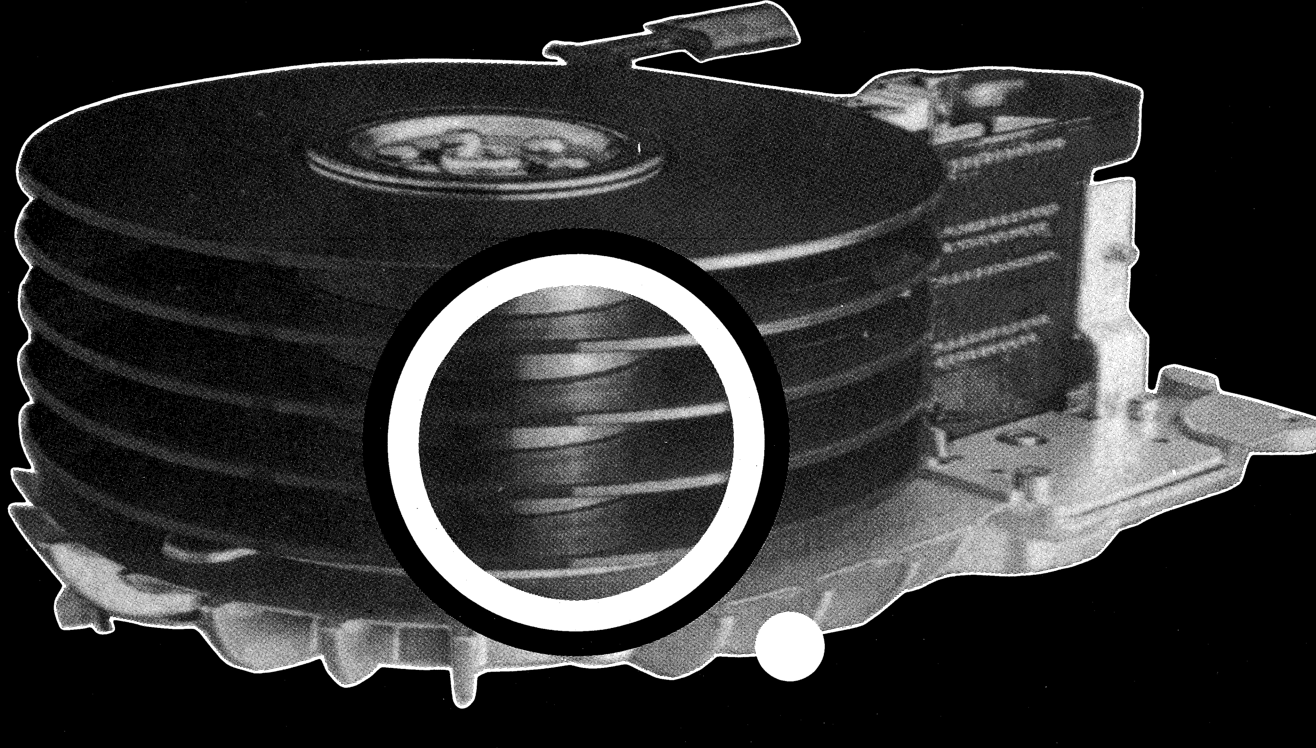
```
_main    endp
;
; void bubble(item, count) /* bubble sort procedure */
; char   *item;
 bubble  proc near
    push si
    push di
    push bp
    mov  bp,sp
    dec  sp
    dec  sp
; int    count;
; {
;   register  int  a,b;
;   register  char t;
;
;   for(a=1; a<count; ++a)
    mov  di,1
    jmp  short @6
@5:
;       for(b=count-1; b>=a; --b) {
    mov  si,word ptr [bp+10]
```

특집

디스크 드라이브의 구조

1 하드디스크 드라이브
2 플로피디스크 드라이브

컴퓨터 보조기억장치로서 가장 많이 사용되고 있는 디스크 드라이브는
IBM-PC가 등장하면서 필수장비로 인식되고 있다.
이러한 하드디스크와 플로피디스크의 하드웨어적인 특성과 종류별 차이점
그리고 데이터의 리드/라이트 원리 등에 대해서 자세히 알아본다.





하드디스크 드라이브

이상훈/시게이트 기술과장

하드디스크 드라이브(이하 HDD)는 데이터가 기록되는 판(Platter, Disk)이 딱딱한 물체로 된 모든 디스크 드라이브를 말하며, 헤드와 디스크가 하나의 틀에 들어가 봉해진 형태로 된 것을 윈체스터 디스크 드라이버(Winchester Disk Driver)라 한다. 헤드와 디스크의 어셈블리를 데이터 팩 또는 HDA(Head Disk Assembly)라고 부르는데, 이 HDA가 디스크 드라이브에 고정되어 사용되는 유형을 고정형 윈체스터 디스크 드라이브라고 한다. 현재 가장 일반적으로 상용되고 있는 것이 바로 이것이고, IBM에서는 고정형 디스크 드라이브라고도 부른다.

HDD 기술개발은 매체(Media), 리드/라이트 헤드, 스텝모터의 개발이 그 열쇠가 되고, 이들 개발이 HDD 발전의 역사라 하여도 과언이 아닐 것이다.

또한, HDD는 소형화, 대용량화, 고속 액세스 타임 등을 추구하고 있다. 5.25인치 드라이브가 주종을 이루던 것이 그 성능 및 신뢰성에 바탕을 둔 3.5인치 드라이브로 눈에 띄게 발전해 가고 있다. 기억용량은 보조기억장치의 성능을 좌우하기 때문에 이의 기술 증대를 위한 가능한 모든 기술이 동원되고 있다. 또한 BPI(Bit Per Inch) 및 TPI(Track Per Inch)를 증대시켜 소형화, 대용량화를 추구하고 있다.

액세스 타임은 HDD가 컴퓨터로부터 명령을 받아 헤드가 원위치에서 리드/라이트하고자 하는 데이터가 있는 곳(라이트할 매체의 트랙)까지 도달하는 시간을 의미하는 것이다.

보통 16~85ms가 소요되고 있으나 기술 개발로 액세스 타임이 계속 줄고 있는 실정이다.

HDD의 구조

여기서는 현재 가장 일반적으로 많이 사용되는 시게이트사의 ST225 내부구조를 위주로 설명하겠다. HDD는 그림 1에서 보는 바와 같이 다음의 부분들로 이루어져 있다. 이들에 대해 간략하게 알아본다.

●스핀들 모터(Spindle Motor)

3600RPM(Revolution Per Minute)의 일정한 속도로 매체를 회전시키며, 전원이 연결되면 리드/라이트 추적에 관계없이 항상 회전한다.

이때 정확한 회전율의 제어가 신뢰성 있는 데이터의 리드/라이트를 위해 필요하다.

●디스크

데이터가 기록되는 판으로 Al판에 자화층이 형성되어 있다. 용량에 따라서 디스크의 장수가 늘어나게 된다.

●리드/라이트 헤드

콘트롤러(Controller)에서 기록전류를 받아 이에 상응하게 디스크의 자화층에 데이터를 기록하고 판독한다.

디스크와 리드/라이트 헤드의 관계는 디스크의 성능을 좌우하는 매우 중요한 요소가 된다.

●스텝 모터(Step Motor)

포지셔닝(Positioning) 모터 중에서 대표적인 것이 스텝모터와 리니어모터이며 HDD에서는 스텝모터가 일반적으로 쓰인다. 스텝모터는 헤드를 이동하는 원동력이 되며, 콘트롤러에서 스텝펄스를 받아 그만큼의 트랙을 이동하게 한다. 따라서 스텝모터의 성능이 액세스 타임을 좌우하는 중요한 요소가 된다.

●전자회로 기판

이 전자회로기판(PCB)에는 스핀들, 스텝 모터를 정밀하게 구동시키기 위한 전자회로가 들어 있으며, 데이터를 정밀하게 판독하기 위한 리드/라이트 회로와 노이즈를 제거하기 위한 필터, 그리고 컴퓨터와의 인터페이스회로 및 오토파킹(Auto Parking)을 위한 제어회로가 내장되어 있다.

HDD의 동작

●플라잉 하이트

HDD는 FDD(Floppy Disk Driver)와 달리 매우 빠른 속도로 회전(3600 RPM정도)하고 디스크의 표면도 매우 딱딱하다. 만일 HDD가 FDD처럼 데이터를 리드/라이트 할 때, 디스크의 표면에 접촉을 하면, 디스크의 고속 회전력과 딱딱한 표면에 의해서 헤드나 디스크가 쉽게 손상되어 버린다. 그래서 HDD에서는 헤드의 구조를 특수하게 설계하여 디스크가 회전시 생기는 바람에 의해서 헤드를 디스크로부터 일정한 간격을 두고 뜨게 한다. 이 높이를 플라밍 하이트(Flying Height)라고 하는데 이는 일반적으로 15에서 20 마이크로인치 정도가 된다. 이 높이는 담배입자 연기보다 작다.

이와 같은 이유로 해서 HDA 내부는 매우 깨끗한 상태의 공기 청정도가 유지되어야 한다. 따라서 HDD의 생산은 이와 같은 조건을 만족하는 Class 100의 크린 룸에서 이루어진다. Class 100이란 1입방 피트내에 직경이 20 마이크로인치 이하인 먼지 입자가 100개 내로 있는 상태를 말한다. 그림 2는 리드/라이트 헤드 스택을 뒤집어 놓은 상태의 그림이다.

●헤드의 이동방식

(1) 오픈 루프

일반적으로 스텝모터를 사용하며 Rack–Pinion 또는 밴드에 의해서 스텝모터의 회전력을 헤드의 이동으로 전환한다. 스텝모터는 입력되는 전기 펄스 수에 상응하는 각만큼 회전하고 이 운동이 헤드의 운동으로 전달된다. 스텝모터를 사용시 헤드 운동의 정확도는 스텝모터의 스텝 크기에 의해 좌우된다. 스텝 수를 증가시키면 정밀도는 증가하지만 액세스 타임이 증가하고 비용이 많이 들게 된다. 그러므로 일반적으로 스텝모터는 저용량, 저속의 드라이브에 사용되는경향이 많다. 하지만 Pseudo–Feedback 방법을 사용한 멀티–페이서 스텝모터와 같은 기술의 향상은 스텝모터를 사용한 고성능의 드라이브를 만들어 낼 수 있게 한다(그림 3 참조).

(2) 클로즈 루프

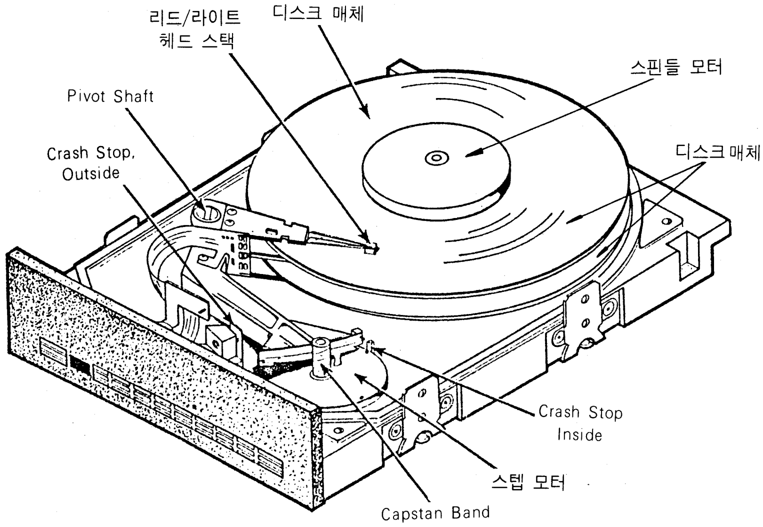
음성코일(Voice Coil)을 사용하며 서보방식이라고도 한다. 음성코일의 동작방식은 라디오에 사용하는 스피커의 원리와 동일하며, 서보패턴을 읽어서 그 신호가 다시 음성코일로 피드백되기 때문에 클로즈 루프라고 한다. 서보패턴은 드라이브 생산시 디스크에 기록되며, 특정한 디스크의 한면을 이 데이터를 위해 사용하느냐의 유무에 따라 데디케이트(Dedicated) 방식과 임베디드(Embedded) 방식이 있다. 서보패턴은 드라이브가 원하는 [illegible] 치에 정확하게 도착할 수 있도 [illegible] 치에 관한 정보를 갖고 있다. [illegible] 루프를 사용하는 경우 고속의 [illegible]드 [illegible]이 가능하며, 스텝모터보다 더욱 [illegible]한 이동을 할 수 있어서 고성능의 [illegible]라이브에 사용된다(그림 4 참조).

기록 및 판독의 원리

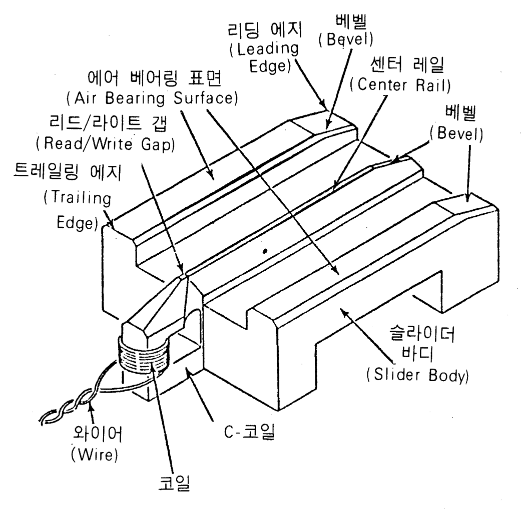
●기록

자성체의 히스테리시스(Hysteresis) 곡선을 이용한 것으로 전류의 크기에

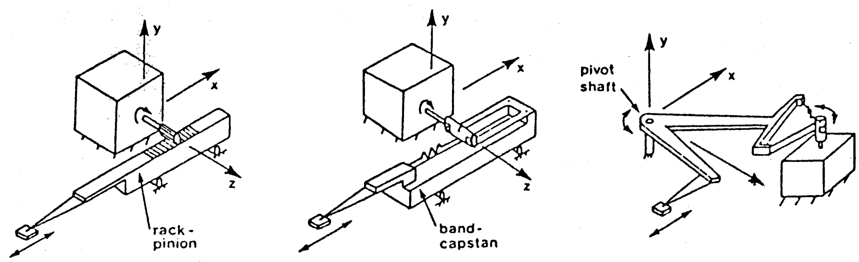
〔그림 1〕 HDD의 외관(ST-225)



〔그림 2〕 리드/라이트 헤드 스택

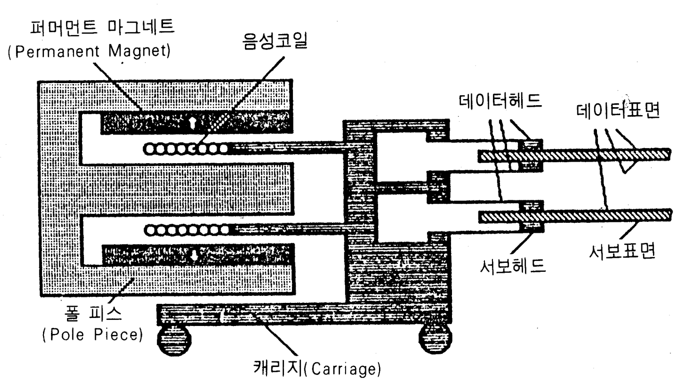


〔그림 3〕 오픈 루프 방식의 헤드 이용

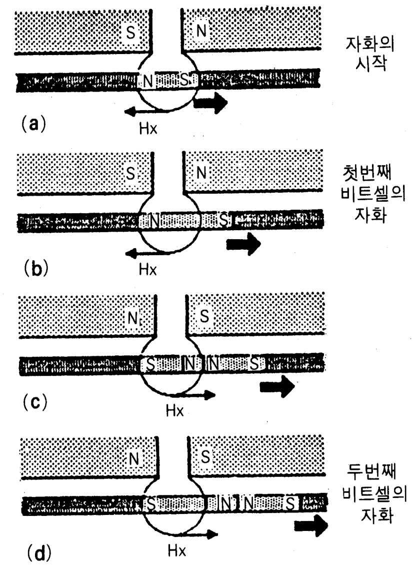


(a) 리니어 Rack-Pinion (b) 리니어 Band-Capstan (c) 로타리 Band-Capstan

〔그림 4〕 클로즈 루프 방식의 헤드 이용



〔그림 5〕 자화 매체에서 기록과정



비례하는 자계가 일정한 범위를 넘어서면 자속을 포화상태로 만들게 된다. 이 포화상태의 극성이 자극을 결정하며, 자성체의 자극이 배영되는 방법에 의해서 데이터가 "1" 또는 "0"으로 결정된다. 즉, 기록전류의 방향이 자속의 변화를 가져오고 자속의 방향에 의해서 데이터는 기록된다(그림 5 참조).

●판독

시간의 변화에 따라 변동하는 자속은 전압을 유발시키는 원리를 이용한 것으로 헤드 아래에서 변화하는 자속의 방향은 리드 백(Read Back) 전압을 만들어 낸다. 이 전압은 프리 앰프를 통해 증폭되고 증폭된 전압은 미분기와 제로 클로싱 디텍터를 통해 전압의 극점을 찾아내고 이 극점이 데이터로 읽혀진다. 이외에 잡음의 제거를 위한 회로를 통과한다.

●수평기록과 수직기록

현재는 주로 자성체가 수평이 되는 방향으로 기록하고 있으나 데이터밀도를 높이기 위해 자성체가 수직으로 자화되는 수직기록방식도 있다.

매체

●산화 매체

Al로 된 기판위에 자화가 되는 $\gamma-Fe_2O_3$ 입자와 공간을 채우는 Al_2O_3 그리고 이 두개의 성분을 결합해주는 열가소성 바인더(Thermoplastic Binder)로 코팅된 층이 있고, 이 위에 윤활층이 놓이게 된다. $\gamma-Fe_2O_3$ 입자는 헤드에서 생기는 자계에 의해 방향이 정해지며, 이 방향은 "1" 또는 "0"으로 지시한다. Al_2O_3는 매체의 마모방지를 위해 사용되고 맨 위의 윤활층은 헤드가 디스크의 회전에 의해 뜨기전에 디스크와의 접촉시 생기는 마찰을 방지하기 위한 것이다.

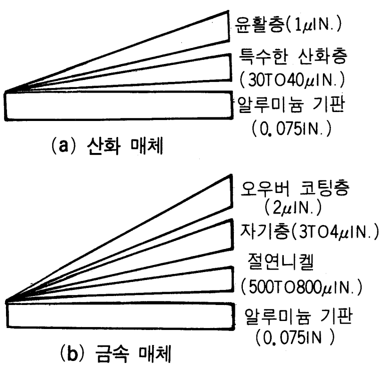
●금속의 얇은 필름 매체

산화 매체와 마찬가지로 이 금속의 얇은 필름 매체(Metal Thin Film Media)는 Al 기판위에 니켈과 같은 금속성분을 사용하여 매체의 견고성을 증가시키기 위한 층(Under Coat)을 만들고, 이 위에 코발트, 니켈, 크롬과 같은 금속 또는 이들의 합금을 자화되는 층으로 구성한 것이다. 맨 위층은 산화매체처럼 오버코트라는 층이 있어 디스크와의 접촉시 마찰을 막아주며, 부식방지의 역할도 한다.

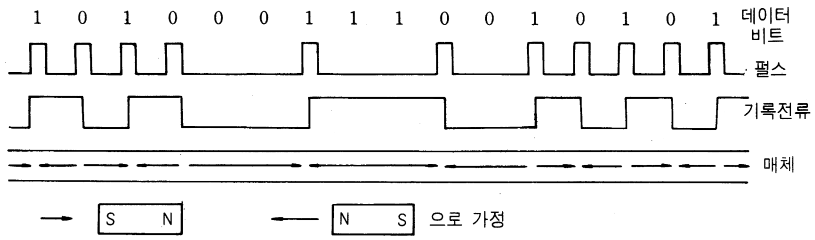
●산화 매체와 금속의 얇은 필름 매체의 비교

산화매체의 자화층은 금속의 얇은 필름매체에 비해 두껍기 때문에 리드 전압의 특성이 떨어진다. 그리고 금속은 산화에 비해 견고하며 자화물질이 같은 영역내에 훨씬 많이 포함되어 있어서 비트 밀도를 증가시킬 수 있다. 이것은 금속에서 산화와 같이 필터나 바인더(Binder)가 필요하지 않기 때문이다. 이러한 금속의 얇은 필름 매체의

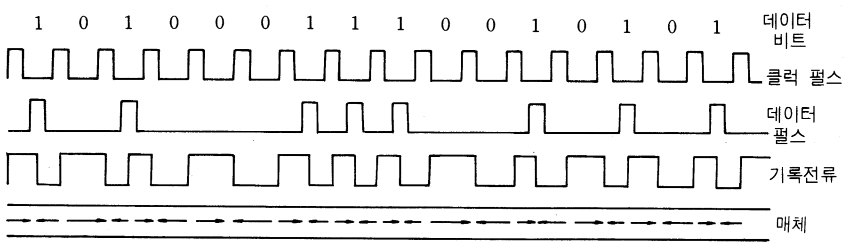
〔그림 6〕 산화·금속 매체의 비교



〔그림 7〕 NRZ 자속변화의 표기



〔그림 8〕 NRZ의 예를 FM방식으로 표기



장점에 의해서 고성능, 고용량의 드라이브에 사용된다(그림 6 참조).

데이터 인코딩

디스크 드라이브에 기록되는 데이터는 자속변화가 선형적으로 배열된 것이다. 이것은 8비트의 데이터가 8개의 자속변화(Flux Transition)에 의해 표현된다는 것이다. 그러나 항상 그런 경우는 아니며, 어떤 경우는 두개의 자속 변화가 1개의 비트를 의미하기도 하고 1개의 자속변화가 1.5개의 비트를 표현하기도 한다. 데이터 인코딩의 목적은 최소의 자속변화로 최대의 데이터를 표현하는데 있다.

●NRZ

NRZ(Non Return to Zero)는 초기의 드라이브에 사용되었던 인코딩방식이다. 그림 7에서 보는 바와 같이 "1"에서 "0", "0"에서 "1"로 될 때 자속변화를 비트 셀 중간에서 갖는다.

●FM

이 FM(Frequency Modulation)은 단밀도 FDD에서 시작된 인코딩방식이다. 각 비트 셀의 시작되는 부분에 클럭을 갖고 "1"인 경우 비트 셀의 중간에서 자속변화를 갖는다.

NRZ의 예를 FM으로 표시한 예는 그림 8과 같다.

●MFM

이 MFM(Modified Frequency Modulation) 이 현재 가장 일반적으로 사용되는 인코딩방식이다. FM과 같은 클럭 펄스가 비트 셀의 시작마다 필요하지 않고 연속된 두개의 "0"사이에 클럭 펄스를 갖는다. 데이터는 FM과 같이 "1"인 경우 비트 셀의 중간에서 자속변화를 한다.

FM과 MFM을 비교해보면 FM에서는 자속변화가 24번이며, MFM은 11번이 된다. 자속변화를 하는데 필요한 공간을 FM과 MFM에서 갖게 하면 MFM은 FM의 반정도의 매체영역을 차지한다. 즉, 같은 A395H라는 데이터를 매체에 기록할 때 MFM은 FM방식보다 사용되는 매체는 반에 불과할 정도를 작게 차지한다. 그래서 MFM이 FM 과 같은 매체영역을 갖는다면 2배의 밀도를 갖게 기록할 수 있다. 따라서 보통 MFM을 FM과 비교하여 더블밀도 레코딩방식이라고 부른다.

●RLL

RLL(Run Length Limited)에는 여러가지 종류의 인코딩방식이 있으며, 가장 통상적으로 사용하는 것이 RLL(2, 7) 인코딩방식이다. 여기서 2, 7이라는 숫자는 연속할 수 있는 최소의 "0"의 숫자와 연속할 수 있는 최대의 "0"의 숫자를 의미한다.

RLL은 다음의 표 1에서 보는 바와 같이 데이터를 변환하여 사용한다.

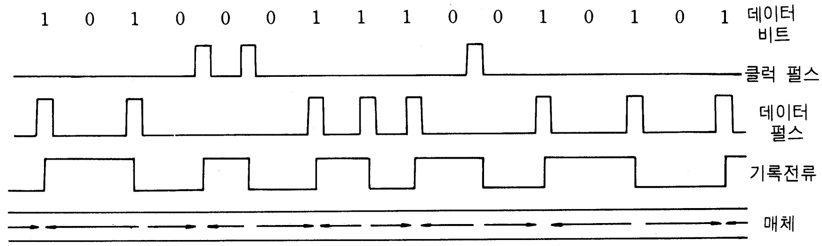
예로써 A395H=1010001110010101B 는 RLL 코드로 바뀌면 처음의 10은 0100으로 또 그 다음의 10은 0100으로, 0011은 00001000으로 전환되어 결국은 A395H=01000100000010000-1001001000100 으로 전환된다. 이것에서 보면 데이터가 코드로 전환되었을 때 연속할 수 있는 최소의 "0"과 최대의 "0"의 수를 쉽게 알 수 있다.

RLL에서 자속변화는 데이터가 "1"인 경우 같게 된다. 즉 바뀔 때 생기게 되므로 A395H의 예를 보면 그림 10에서 보는 바와 같이 7개의 자속변화를 갖는다. 반면에 MFM은 11개의 자속변화를 갖는다. 이것은 FM과 MFM의 관계에서 MFM이 두배의 기록밀도를 갖게된 것처럼 RLL(2, 7)도 MFM에

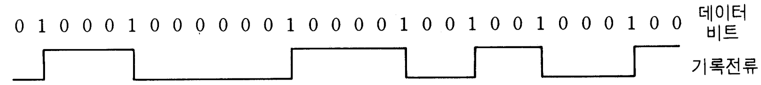
〔표 1〕 데이터를 RLL방식으로 표기

데이터 비트	코 드
0 0 0	0 0 0 1 0 0
1 0	0 1 0 0
0 1 0	1 0 0 1 0 0
0 0 1 0	0 0 1 0 0 1 0 0
1 1	1 0 0 0
0 1 1	0 0 1 0 0 0
0 0 1 1	0 0 0 0 1 0 0 0

〔그림 9〕 NRZ의 예를 MFM방식으로 표기



〔그림 10〕 A395H를 RLL방식으로 표기한 기록전류



비해 약 두배 정도의 기록밀도를 갖는다.

●ARLL

현재 콘트롤러를 내장한 드라이브에서 이 방식을 써서 인코딩하는 제품이 나오고 있다. 이것은 MFM에서 17섹터 / 트랙(512 바이트 / 섹터)으로 포맷하고 RLL에서 26섹터 / 트랙(512 바이트 / 섹터)으로 포맷하는 반면 ARLL (Advanced RLL)은 34 섹터 / 트랙이 가능하다.

즉, RLL에 비해 50%정도의 데이터 밀도를 높일 수 있다.

●존비트 기록

현재 대부분 사용되고 있는 레코딩 방식에는 바깥쪽 실린더와 안쪽 실린더의 비트밀도는 바깥쪽 실린더보다는 안쪽 실린더가 훨씬 비트밀도가 높게 된다. 그러나 존(Zone) 비트 레코딩에서는 드라이브의 실린더를 몇개의 존으로 구분하여 바깥쪽 존에는 안쪽 존보다 더 많은 비트를 기록하게 한다. 그래서 드라이브 매체의 활용을 매우 유용하게 할 수 있고 데이터밀도는 급격히 증가한다. 이 방식은 보통 콘트롤러를 포함하는 드라이브에서 사용될 수 있다

인터페이스

현재 20MB~250MB 용량의 HDD가 PC와 인터페이스되어 사용중에 있다. 이 인터페이스 종류에는 크게 4가지로 분류할 수 있다.대략적인 인터페이스 방법은 그림 11에 나타나 있다.

●ST412 / 506

요즘 가장 많이 사용하는 인터페이스이다. 데이터의 판독(리드)에 매우 중요한 역할을 하는 데이터 분리를 드라이브 내에 포함하지 않고 MFM, RLL 레코딩 방식에 따라 5MB/s, 7.5MB/s의 전송율을 갖는다.

●ESDI

이 방식(Enhanced Small Device Interface) 은 SMD라는 인터페이스와 ST412 / 506을 조합한 것으로 데이터 분리가 드라이브내에 내고 있고 10MB / s의 전송율을 갖으며 드라이브에서 나오는 데이터는 NRZ이다.

●AT 버스 & XT 버스

PC / AT 버스와 XT 버스의 신호에 약간의 수정을 가하면 직접 AT와 XT에 연결할 수 있는 인터페이스 이다. 콘트롤러가 드라이브에 포함되어 있다.

●SCSI

8대까지의 디바이스를 연결할 수 있고 드라이브뿐 아니라 프린터, 테이프 드라이브 등도 이 인터페이스에 연결하여 사용할 수 있다. 이 SCSI(Small Computer System Interface) 인터페이스는 멀티 유저용으로 사용하기 좋은 장점을 갖고 있다. 즉, 아비트레이션(Arbitration이라는 타임 프레임과 선택이라는 타임 프렘임을 통해 버스를 사용할 디바이스가 연결이 되고 인터페이스를 사용하지않는 순간에 디스코넥트(Disconnect) 같은 명령을 통해 잠시동안 다른 디바이스에게 버스를 사용하게 한 후 다시 리코넥트(Reconnect) 명령을 통해 재연결할 수 있다. 드라이브는 콘트롤러를 포함하여 12MB / s의 전송률을 갖는다.

사용시 유의점

여기서는 하드디스크를 사용시 생기는 문제점을 해결하는 매우 기본적인 방법을 설명한다.

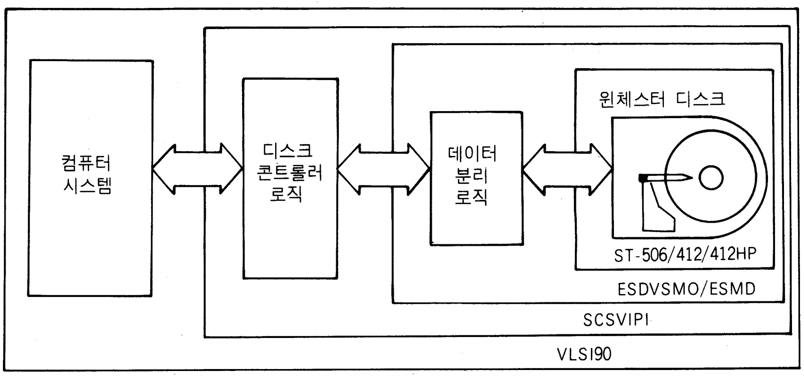
(1) 기본적인 점검사항

드라이브를 설치한 후 혹은 드라이브를 설치하기 전에 다음과 같은 사항을 점검해본다.

① 전원케이블이 드라이브에 올바르게 연결되었는지 점검한다.

② SCSI 드라이브가 아닌 드라이브에서는 20핀의 데이터 케이블과 34핀

〔그림 11〕 인터페이스 방법



의 콘트롤러 케이블을 갖고 있는데 케이블의 1번 핀이 올바르게 연결되어 있는지 확인한다.

③ 드라이브가 시스템에서 첫번째 드라이브로서 사용된다면 드라이브 선택 핀이 올바르게 선택되어 있는지 확인한다.

④ 드라이브 하나가 사용될 때 드라이브에 있는 터미네이팅(Terminating) 레지스터 팩이 설치되어 있는지 점검한다. 만일 두개의 드라이브를 사용하는 경우 첫번째 드라이브의 터미네이팅 레지스터 팩을 제거하고 두번째 드라이브의 터미네이팅 레지스터 팩은 그대로 둔다.

(2) 에러 메시지

● 1701

이것은 시스템이 드라이브를 전혀 인식하지 못하는 경우이거나 드라이브가 전혀 동작하지 않는 경우 (Drive Not Ready)의 상태이다. 점검할 부분은 다음과 같다.

① 드라이브의 전원 케이블이 올바르게 연결되어 있는지 확인한다.

② 24핀, 34핀 케이블 그리고 터미네이팅 레지스터 팩이 제대로 연결되었는지 확인한다.

③ 콘트롤러 카드의 점퍼(Jumper) 위치들이 올바르게 되어있는지 확인한다.

④ 드라이브의 스핀들모터가 회전하는지 확인한다.

● 부트 페일

① 파티션(Partition)이 되었는지 확인한다.

② 시스템 화일이 올바르게 전송되는지 확인한다.

③ 케이블이 혹시 잘못 연결되어 있지 않는지 확인한다.

● 리드 / 라이트 에러

① 포맷시 하드디스크의 매체 디펙트 맵을 입력시켰는지 확인한다.

② 전원 케이블이 드라이브에 정확하게 연결되어 있는지 확인한다.

③ 만일 HDD가 MFM용이라면 콘트롤러 카드는 MFM용이어야 한다.

앞에 설명한 사항은 매우 기본적이며 정확한 불량분석은 경험이 있는 전문가에게 의뢰하여야 한다.

일반적으로 복구가능한 불량을 유저가 실수로 복구불능을 가져오게 하는 경우가 비일비재하므로 매우 주의하여 드라이브의 문제점을 대처하여야 한다.

(3) 적당한 공기순환과 설치방법

HDD는 동작중에 열을 발산하는 부분을 많이 갖고 있다. 그러므로 드라이브의 사용환경은 이러한 열을 분산시키기 쉽게 되어 있어야 한다.

또한 드라이브를 시스템에 고정시 스크류(Screw)를 사용하여 조이는 경우가 있는데 이때에도 드라이브 자체에 영향을 주지않도록 조심해야 한다.

(4) 정전기 방지와 취급

전자부품을 갖는 모든 기기에서와 마찬가지로 HDD에서도 정전기에 의해서 드라이브의 PCB가 손상하지 않도록 주의해야 한다. 일반적으로 드라이브를 구입시 드라이브는 정전기 방지용 비닐봉지에 들어 있게 된다. 드라이브를 설치할 때 맨손으로 드라이브를 만지는 것보다 정전기방지 비닐위에 놓고 운반하며, 특히 맨손으로 드라이브의 PCB를 만지지 않도록 해야한다.

또한 드라이브는 외부의 충격에 매우 민감하게 설계되어 있다. 이것은 플라잉 하이트와 밀접한 관계가 있는데, 이것은 동작시와 동작하지 않을 때 모두 해당된다.

헤드는 드라이브내에서 스핀들모터가 회전하는 바람에 의해서 뜨기 때문에 매우 가벼운 축을 갖고 있다.

이것은 외부에서 드라이브에 충격을 줄때 헤드와 디스크의 표면이 부딪혀서 헤드나 혹은 디스크가 손상될 우려가 있다.

또한 동작중에 드라이브에 충격을 주는 것은 매우 치명적이다. 이것은 플라밍 하이트가 그 이유를 정확히 설명해 준다.

현재 사용되고 있는 드라이브 중에는 전원을 끄게 되면 헤드가 데이터 영역에서 벗어나 자동적으로 쉬핑존 또는 파킹존이라고 불리는 영역으로 이동하는 것도 있다.

이것을 오토파킹(Auto Parking)이라고 하는데 이러한 기능을 가진 드라이브는 별문제가 안되지만 그렇치 않는 드라이브에서는 드라이브가 동작후 전원을 끄면 리드 / 라이트하던 곳에 헤드가 그대로 머물러 있게 된다.

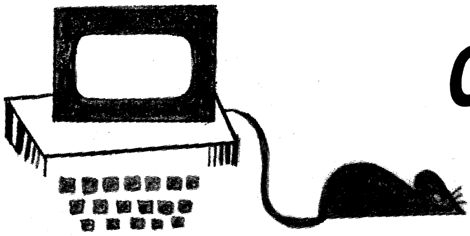
이때 충격을 받게 되면 데이터를 손실하는 결과를 초래할 수 있으므로 소프트웨어를 사용하여 헤드를 쉬핑존(Shipping Zone)으로 이동 시켜야 한다.

맺음말

HDD를 사용하면서 매우 많은 사람들이 HDD가 매우 안전하리라는 생각을 한다. 물론 드라이브의 전자적, 기계적 발전은 더욱더 신뢰할 수 있는 드라이브를 생산해낸다. 그러나 드라이브의 용량이 커질수록 위험부담은 커지며, 불시에 드라이브가 손상을 입는 경우 그 손해를 상상할 수 없다. 그러므로 드라이브 사용시 중요한 데이터는 백업이 필요하며, 항상 드라이브에 외부적인 충격을 주지 않게 조심하여야 한다.

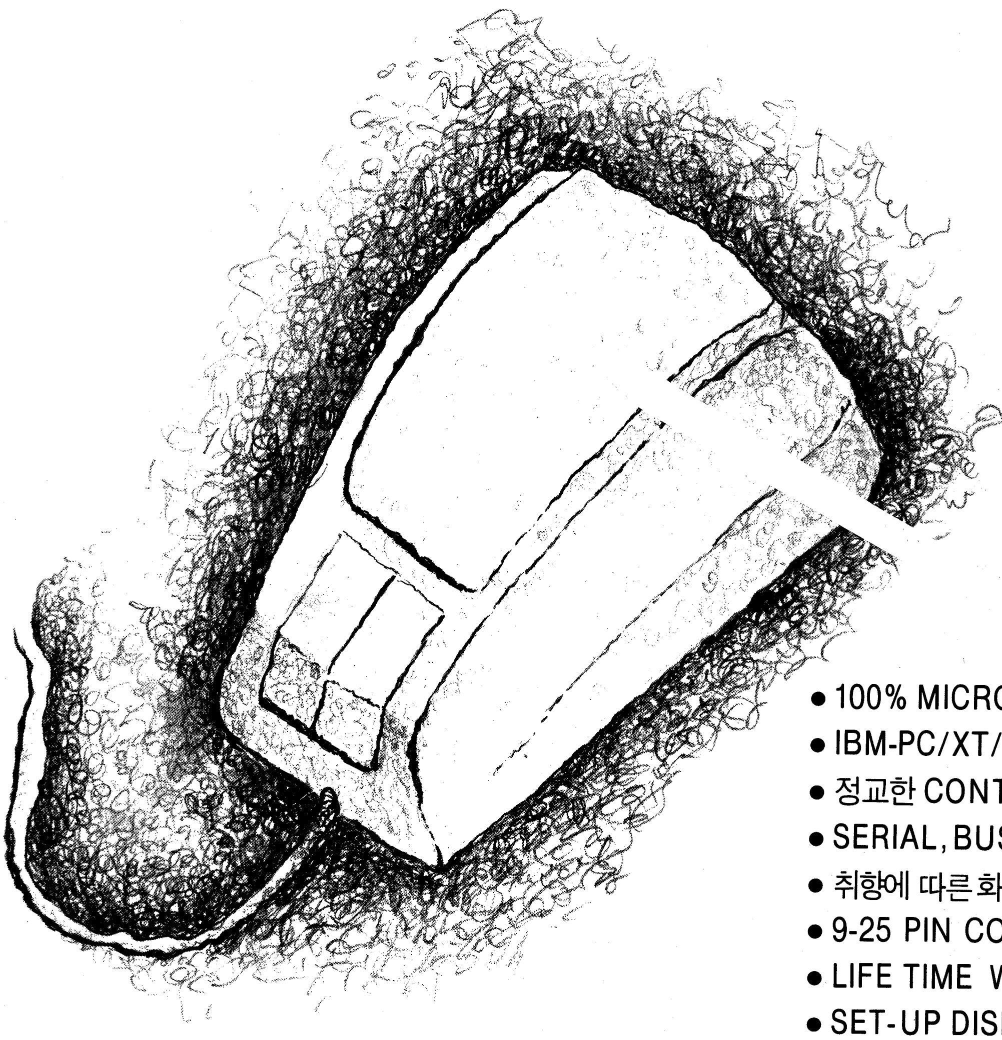
많은 드라이브 제조회사에서 드라이브의 충격으로부터 데이터보호를 위해 파워다운시 헤드를 자동적으로 데이터영역 밖으로 보내는 오토 파킹 기능을 드라이브에 갖도록 하고 있다.

그러나 이러한 모든 장점을 가지고 있다할지라도 각자 용도에 맞는 시스템을 선택하는 것이 가장 바람직하다. PC



이제 Mouse는 PC의 필수품

미국에서 선풍적으로 판매되고 있는
VATEK사의 COLOR-MOUSE ™
드디어 국내 판매 개시



- 100% MICROSOFT CO... TIBLE
- IBM-PC/XT/AT,PS/2 AND C... TIBLE
- 정교한 CONTROL(250 DPI) 수평/수직
- SERIAL, BUS TYPE
- 취향에 따른 화려한 색상
- 9-25 PIN CONNECTOR 포함
- LIFE TIME WARRANTY
- SET-UP DISK AND DRIVER

VATEK
Valuable Technologies

VATEK USA INC.
4636 Macbeth Court
Fremont. CA94536
TEL:415-794-6721
FAX:415-792-5932

한 국 바 테 크

서울·강남구 역삼동 769-15
한종BLDG.
TEL : 566-1108 (代)~9
568-5897
557-5087
FAX : 555-4146

대리점 모집

- PC 및 주변기기 판매영업 하시는 분.
- 전국대도시 모두 해당



●특집 2●

플로피디스크 드라이브

김학준/동양정밀공업(주) 해외사업본부 HDD사업부

1972년 8인치 단면디스크를 사용한 IBM 3740 데이터 엔트리 시스템이 발표된 후 1976년 미국 슈가트사가 5.25인치 미니 FDD(Floppy Disk Drive)를 개발, 발매한 것이 FDD가 본격적으로 등장하게 된 시초였다. 그후 1980년대 들어 일본의 소니사가 미국에서 3.5인치 마이크로 FDD를 사용한 영문 워드프로세서를 발표하여 현재의 3.5인치 마이크로 FDD의 시초가 되었다.

당초 이들 FDD는 시스템의 고성능화에 대응하여 기록방식이 편밀도에서 배밀도로, 단면에서 양면으로 그리고 트랙밀도를 증가시키는 등 기억용량을 배가시켰다. 특히 현재시스템의 소형경량화, 저전력소비화 등에 대응하여 FDD는 박형화되어 개발되고 +5V 단일전원에 프레임의 높이가 1인치이며, 300g 정도인 FDD가 발표되고 있으며 앞으로도 시스템의 발전추세에 따라 FDD는 끊임없이 가속적인 변화를 추구할 것이다.

따라서 본고에서는 독자의 FDD에 대한 이해를 돕고자 한다.

FDD사용시 많은 참고가 되길 바란다.

FDD의 종류 및 구조

(1) 8인치 FDD

1972년에 발표된 FDD는 편밀도 단면으로 데이터의 기록용량이 250k바이트 정도였으나, 그 후에 배밀도 양면 등의 기술이 개발되어 8인치 FDD의 용량을 배가시켜 왔다.

프레임의 두께 또한 1/2로 박형화되었다.

그러나 현재로선 동일한 기록용량을 가진 혹은 그이상의 용량을 가진 5.25인치나 3.5인치 FDD의 개발로 그 가치는 감소되었지만 8인치 FDD의 가장 큰 특징인 IBM 시스템과의 호환성 때문에 굳이 호환성이 요구되지 않는 곳에서는 고밀도 5.25인치 FDD를 사용할지라도 조금의 데이터호환을 고려해야 하는 시스템일 경우에는 아직도 이 8인치 FDD를 필요로 해 당분간은 계속 사용될 것으로 보인다.

단, 주의해야 할 것은 IBM 시스템과 호환을이루는 부분은 디스크크기와 포맷형태 뿐이고 인터페이스와 사이즈면은 각 메이커마다 차이가 있다는 점이다.

(2) 5.25인치 FDD

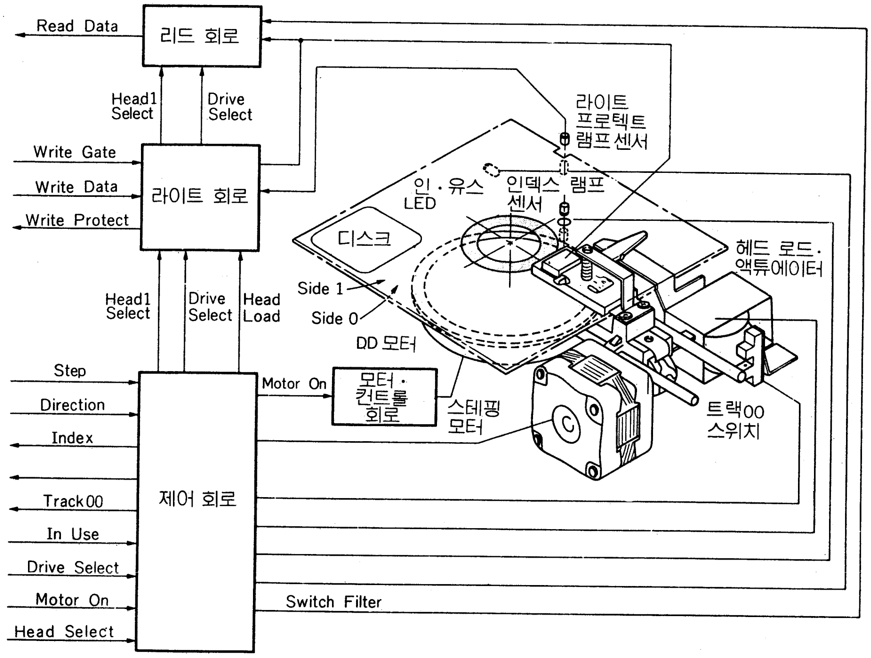
미국 슈가트사에서 발표된 5.25인치 FDD는 8인치 FDD와 같은 트랙밀도로 1인치에 48트랙이 들어가는 것인데, 트랙수는 35이고 단면 편밀도로 기억용량은 포맷없이 109k바이트이며 포맷하여 실제 사용되는 용량은 82k바이트였다.

그후 계속적인 기억용량의 증대화를 도모함에 따라 여러가지 신기술 및 새로운 소재가 개발되어 그 기억용량은 포맷없이 0.5M바이트나 1M바이트, 1.6M바이트 등으로 증가하였고 단면에서 양면과 양면배밀도로 발전하였다.

또한 트랙간 3ms의 고속 액세스와 DC브러시레스(Brushless) 다이렉트 드라이브 모터 채용으로 무보수화하여 신뢰성을 향상시켰다. 특히 전력의 저소비화 등으로 고성능화를 이룩하였는데 IBM PC가 발표되면서 그와 같은 고성능화에 대한 중요성은 더욱 부각되었다.

프레임 사이즈는 8인치 FDD와 마찬가지로 미국 슈가트사의 규격에 디스크, 디바이스, 인터페이스 부분이 호환성을 이룬다.(그림 1 참조)

〔그림 1〕5.25인치 FDD의 내부 구성도



(3) 3.5인치 FDD

3.5인치 FDD는 마이크로 FDD에 속하는 것이다. 마이크로 FDD는 8인치나 5.25인치 FDD에 비해 보다 소형이고 저가격 시스템용으로 지름이 3~4인치인 디스크를 사용한 FDD를 말한다. 이들은 3인치, 3.25인치, 3.5인치, 4인치 등 4규격으로 나뉘어져 있는데 각기 사이즈가 다르듯 규격도 달라서 8인치나 5.25인치 FDD와 같이 서로 호환성은 이루지 못하고 있다. 단지 각 개별로 미국의 ANSI규격으로 제안되고 있다.

이중에서도 3.5인치 FDD는 83년에 MIC(Microfloppy Industry Committee : 미국의 슈가트, 탠덤(Tandom) 등의 메이커들로 결성된 그룹)와 소니가 규격을 정리하여 ANSI에 제안하였다. 이 때 단면으로 80 또는 40트랙과 회전속도는 600rpm 또는 300rpm이었다.

또한 트랙당 기록용량은 5.25인치와 같게 하고 300rpm의 FDD이면 5.25인치 FDD와의 포맷 호환성이 있다.

또한 이 3.5인치는 원래 80트랙으로 설계되어 배트랙밀도 96TPI 5.25인치 FDD의 대치용으로 사용되기 시작하는 등 최근들어 수요량이 급격히 증가하고 있으며, 특히 IBM PS/2 모델이 발표되면서 그 효용성이 높아지고 있다.

최근의 랩탑 컴퓨터 등에선 고속화, 경량화, 저전력 소비화가 요구되어 프레임의 높이가 1인치로 로우 프로파일(Low Profile) FDD가 발표되면서 그 가치를 한층 증가시키고 있다.

또한 89년도 이후에는 3.5인치 FDD의 수요량이 5.25인치 FDD 수요량을 초과할 것으로 예상된다.

그림 2에 3.5인치 FDD의 내부 구성도를 보였다.

이와같이 지금까지 설명한 8인치나 5.25인치, 3.5인치는 종래부터 주된 관심의 대상이 되었던 FDD들이다. 그리고 그외에도 비주류의 FDD로는 초소형과 2~2.5인치 정도의 FDD 및 수직자기기록 방식에 의한 FDD들이 있다.

FDD 메카니즘 및 회전부분

보조기억장치란 주기억장치를 대신해서 데이터를 저장 재생하는 것을 목적으로 하는 장치이다. 이중에서 특히 HDD는 매체와 헤드가 서로 고정되어 있어 상호간의 호환성이 재고될 이유가 없지만 반면에 FDD는 어떤 회사의 드라이브를 이용하여 데이터를 저장했다하더라도 다른 회사의 드라이브를 이용해도 재생할 수 있어야 하기 때문에 서로의 호환성이 유지되어야 한다.

FDD는 사이즈에 관계없이 헤드 및 헤드지지 메카니즘과 헤드위치 정하기 메카니즘, 디스크구동 메카니즘, 디스크클램프 메카니즘, 헤드로드 메카니즘, 디스크보호 메카니즘, 스테이터스 검출부 및 콘트롤 회로로 구성된다.

즉, FDD의 주목적은 주기억장치를 대신하여 데이터를 기록, 재생하는 장치이기 때문에 리드/라이트하는 매체와 매질로써 헤드와 디스크가 필요하며, 이를 원하는 트랙에 헤드를 위치시키기 위한 메카니즘이 스텝모터이고 상대적인 유기전압을 발생시켜 데이터를 리드/라이트하기 위한 스핀들모터, 그밖에 부수적인 기구로 구성되어 있는데 이러한 메카니즘은 전자회로에 의해 제어가 된다. 그러나 FDD는 거의 80%가 메카니즘에 속한다.

또한 드라이브 자체로만은 아무런 기능을 행할 수 없으며 실제 화일을 리드/라이트하기 위해서는 드라이브 콘트롤러가 호스트로부터 리드/라이트/시크(Seek) 등의 명령을 받고 이 받은 명령은 드라이브 인터페이스신호를 이용하여 시크를 원할 경우 스텝과 방향신호(Direction Signal)를 이용하여 헤드를 원하는 트랙에 위치시킨다. 만약 어떤 화일을 리드/라이트하고자 할 경우 라이트 게이트 신호를 True로 하고 데이터를 전송시킨다.

이러한 이유로 드라이브 인터페이스는 실제 드라이브 자체를 제어하기 위한 제어신호와 드라이브의 스테이터스,

즉 드라이브 상태가 리드 / 라이트 / 시크 등의 동작이 가능한가를 콘트롤러에게 알려주는 신호로 구성되어 있다.

그러면 이제부터 우리에게 가장 익숙하고 널리 알려진 5.25인치 FDD의 메카니즘 구성 및 리드 / 라이트회로, 콘트롤회로와 인터페이스 관계에 관해서 알아 보기로 하자.

(1) 마그네틱 헤드

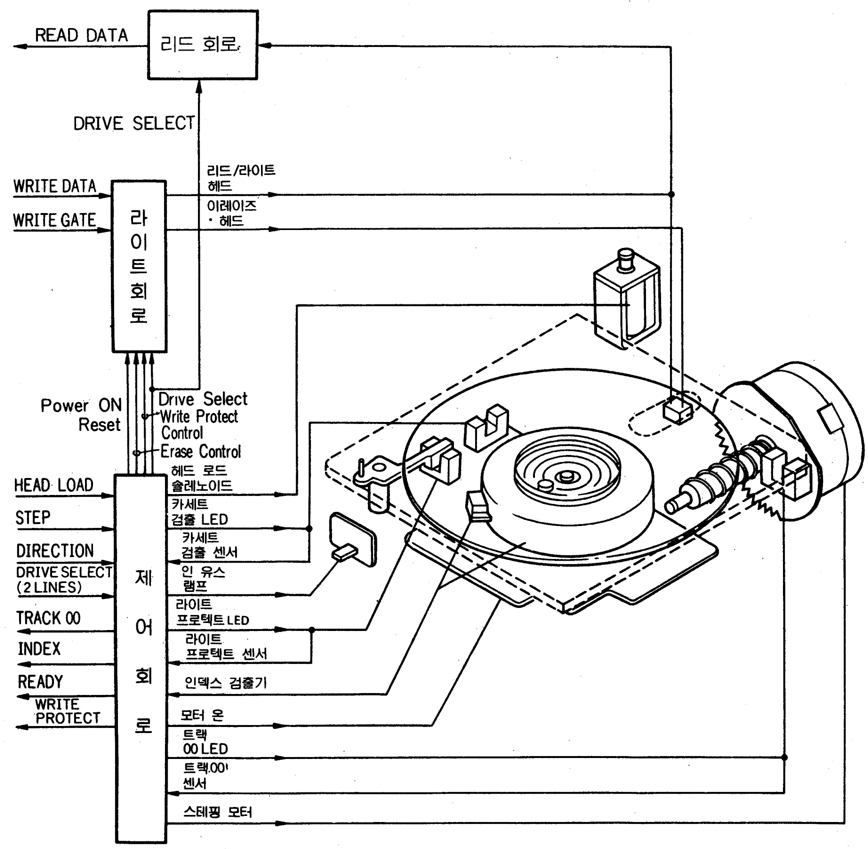
FDD는 데이터의 호환성을 유지하기 위해 트랙피치(48TPI=0.53㎜)보다 작게 데이터를 기록하는데 **그림 3** (a)와 같이 터널소거(Tunnel Erase)라 불리우는 방식이 채택되고 있다.

즉, 실제 데이터폭은 트랙피치의 약 60%정도로, 실제 헤드가 원하는 트랙에 위치할 경우에 항상 OFF 트랙이 발생하고, 이런 OFF 트랙이 발생했다 하더라도 데이터와 데이터간의 호환성(즉, 데이터와 데이터간의 간격)을 유지하기 위해 이레이즈 헤드(Erase Head)가 리드 / 라이트 헤드의 후방 양쪽에 배치되어 있다. 따라서 이 이레이즈 헤드는 불필요한 데이터를 소거하면서 새로운 데이터를 라이트하는 경우나 다른 시스템에서 리드하는 경우에 인접 트랙으로 부터의 영향을 최소화하기 위해 트랙 옆부분을 소거하는 것이다. 그러나 리드 / 라이트 헤드와 이레이즈 헤드간이 서로 떨어져 있으므로 헤드의 ON / OFF시간에 콘트롤회로가 필요한 단점이 있다.

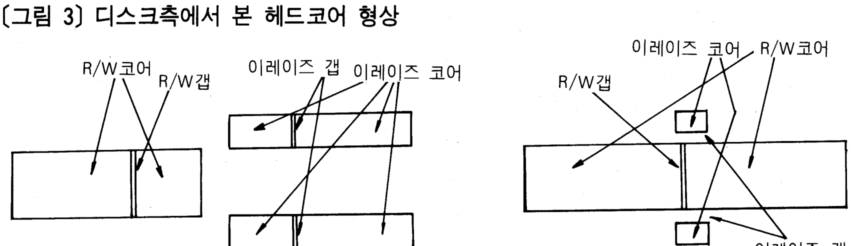
또한 **그림 4**에서와 같이 헤드의 배치가 Side 0와 Side 1에서 서로의 간섭을 작게 하기 위해 48TPI인 경우 4트랙, 96TPI인 경우 8트랙정도 서로 어긋나게 배치되어 있음을 알 수 있다.

즉 그림 4에서 48TPI와 96TPI의 헤드의 배치를 보면 48TPI의 0, 1, 3… 트랙에 대해 96TPI의 0, 2, 4, 6… 트랙위치가 맞도록 설계되어 있다. 따라서 40트랙으로 된 디스크는 80트랙의 FDD로 리드할 수 있다.

〔그림 2〕 3.5인치 FDD의 내부 구성도



〔그림 3〕 디스크측에서 본 헤드코어 형상

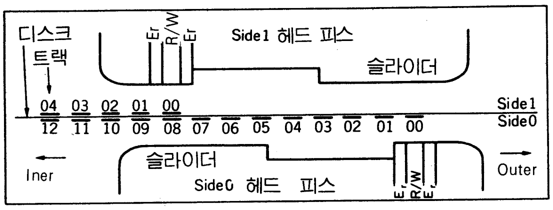


(a) 터널 이레이즈 헤드 (b) 스트래들 헤드

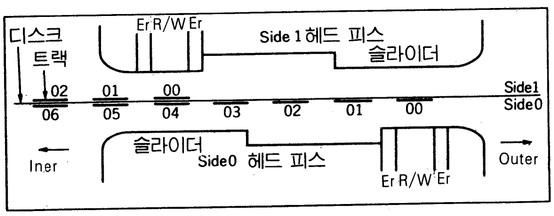
(2) 트랙 시크 메카니즘

헤드 캐리지 드라이브 메카니즘(Head Carriage Drive Mechanism)은 모터의 회전운동을 직선운동으로 바꾸어 주는 스파이럴 캠 포지셔너(Spiral Cam Positioner) 와 리드 스크류 포지셔너(Lead Screw Positioner) 그리고 스틸벨트 포지셔너(Steel Belt Positioner) 등 3가지 방식으로 나눌 수 있다. 이중 헤드구동계의 마찰로 인한 손실이 적어 전력소모가 작고, 고속도로 헤드 이송이 가능하며 정밀도가 높은 스틸벨트방식이 5.25인치 경우에 있어서 가장 널리 사용되고 있다. 그러나 이 스틸 벨트 방식은 8인치나 마이크로 FDD에서도 최고속의 방식으로 꼽히고 있는데 1트랙의 스텝레이트가 3ms이다. 3.5인치 경우에 있어서는 스페이스 및 로드 등으로 리드 스크류 포지셔너가 사용되고 있다.

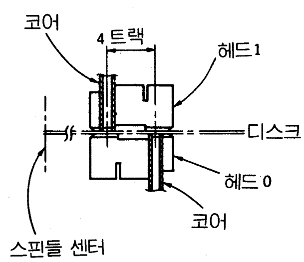
〔그림 4〕 헤드가 트랙 00에 있을 경우의 헤드와 트랙배치



(a) 96TPI



(b) 48TPI



(c) 48TPI 양면 FDD의 헤드와 디스크의 접촉상태

사이드 0과 사이드1의 트랙은 48TPI인 때 4트랙, 96 TPI인 때 8트랙 어긋나 있다.

또한 스텝핑 모터의 위치를 정하는 정밀도가 헤드위치 정도로 직접 나오므로 스텝위치를 결정지어 주기 위해 고정밀도의 스텝모터를 사용하여야 한다. 따라서 헤드를 움직여주는 스텝모터는 96TPI인 경우 1.8°정도 회전하며 서보회로나 메카니즘없이 스텝모터의 정밀도에만 의존한다.

그러나 어떤 경우에 있어 헤드가 원하는 트랙을 시크못할 경우가 발생하는데 이것을 시크에러라 한다. 이런 경우는 스텝모터의 미스스텝(Mis-Step)으로 발생할 수도 있지만 대개의 경우 스텝펄스폭이 너무 짧거나 파워불안 등의 원인으로 발생할 수 있다.

(3) 헤드 로드 메카니즘

HDD는 헤드와 디스크가 0.3μm정도 떨어져 데이터를 리드/라이트 하지만 반면에 FDD는 데이터를 리드하거나 라이트할 때 또는 시크할때 헤드와 디스크가 서로 접촉되어 있기 때문에 디스크에 손상을 줄 수 있다.

따라서 디스크의 수명을 연장하기 위해서는 리드/라이트 동작을 하지 않을 경우에는 헤드를 디스크로부터 분리시켜 놓을 필요가 있다. 이러한 역할을 하는 기구가 헤드로드 메카니즘이다.(그림 5 참조).

즉, 리드/라이트할 때만 헤드를 로드시키고 그외는 헤드를 언로드(unload)시키는 것이다. 그러나 이를 하기 위해서는 솔레노이드가 필요하다. 왜냐하면 헤드가 로드되었다, 언로드되었다 할 때마다 귀에 거슬리는 소리가 날 뿐 아니라 불필요한 전력이 소모되고, 특히 헤드가 디스크에 ON/OFF될때 디스크에 손상을 가져올 수 있기 때문에 요즈음 생산되는 FDD는 리드/라이트할 때만 드라이브 스핀들모터를 구동시킴으로써 헤드로드 메카니즘을 제거했으나 스핀들모터가 ON된 후부터 준비(Ready) 상태가 될 때까지 약500ms 정도 걸리므로 솔레노이드가 장착된 드라이브에 비해 처리속도가 약간 내려간다. 따라서 처리속도를 빠르게 하기 위해서는 솔레노이드가 드라이브에 장착될 필요가 있다.

(4) 디스크회전 메카니즘

이는 디스크클램프 메카니즘으로서 삽입된 디스크를 스핀들에 고정시켜 회전시키는 메카니즘이다. 5.25인치 FDD가 사용하고 있는 것은 디스크에 뚫은 센터홀을 직접 클램프하는 것이다. 즉 헤드를 원하는 위치에 포지셔닝(Positioning) 시키고, 헤드를 이용하여 디스크에 데이터를 라이트하고 서로의 상대적인 운동으로 유기전압을 발생시켜 데이터를 리드한다. 이때 일정한 속도로 디스크를 회전시키고 디스켓의 센터홀(Center Hole)을 정확히 센터링 할 수 있도록 콜릿(Collet)이 설계되어 있다.

이와같이 FDD의 박형화에 따라 이 콜릿의 이동량을 가능한한 크게 하고 이동속도 역시 적당히 콘트롤 할 수 있는 회전레버방식이 늘고 있다.

(5) 스테이터스 검출부

스테이터스 검출부는 FDD의 상태를 보기 위한 것으로 인덱스 센서, 라이트 프로텍트 스위치, 트랙00 스위치, 디스크검출 스위치 및 도어스위치 등의 검출스위치를 갖고 있다. 그러면 이들 각종 스위치에 대해 간략하게 알아본다.

●인덱스 검출기

디스크 회전을 검출하고 트랙의 개시점을 정하기 위한 것으로 디스켓의 비어 있는 인덱스 홀을 통해 LED와 포토 트랜지스터를 이용하여 인덱스를 검출한다.

●트랙00 검출기

디스크의 가장 바깥쪽 트랙을 트랙00이라 하는데 이를 검출하기 위한 스위치가 바로 트랙00 검출기이다. 이 트랙00이 모든 트랙의 기준이 되므로 정확하게 결정, 조정되어야만 한다. 따라서 이러한 조정 및 검사에 얼라인먼트(Alignment) 디스크를 사용해서 중심트랙이 48TPI인 경우 트랙16에서 정확히 조정을 함으로써 트랙00을 결정한다. 헤드가 트랙00에 위치하고 있는 것을 검출하기 위해 LED와 포토트랜지스터가 쌍으로된 포토인터럽터와 트랙00 스톱기(Stopper)가 사용된다.

예를 들어 자동재교정(Auto-Recalibration)이나 시크시 포토인터럽트와 트랙00 스톱기를 이용하여 트랙00을 결정한다.

●라이트 프로텍트 검출기

디스크 자켓에 있는 라이트프로텍트 노치의 유무를 포토인터럽터와 LED를 사용하여 검출하는 스위치로 디스크에 불필요한 데이터가 쓰여지는 것을 막기위한 장치이다.

●디스크인 검출기

디스크가 드라이브세트에 장착될 때 스핀들 모터를 자동으로 ON하기 위한 회로가 디스크인(Disk In)검출기이다.

(6) 메카니즘을 제어하는 전자회로부

각각의 메카니즘을 전기적으로 제어하는 부분으로 이에 대한 자세한 설명은 다음장에서 한다.

전자회로부

FDD을 이루는 전자회로는 리드/라이트회로와 콘트롤회로, 서보회로 등 크게 3개 부분으로 나눌 수 있다.

다음에 설명되는 것은 각 회사제품에 따라 약간씩 차이가 있을 수 있으므로 참고하기 바란다.

(1) 리드/라이트회로

리드/라이트회로는 모드셀렉터, 리드회로, 라이트회로 및 로우전압센서 등으로 구성되어져 있다. 그러면 이들이 어떤 역할을 하는 것인지 알아본다.

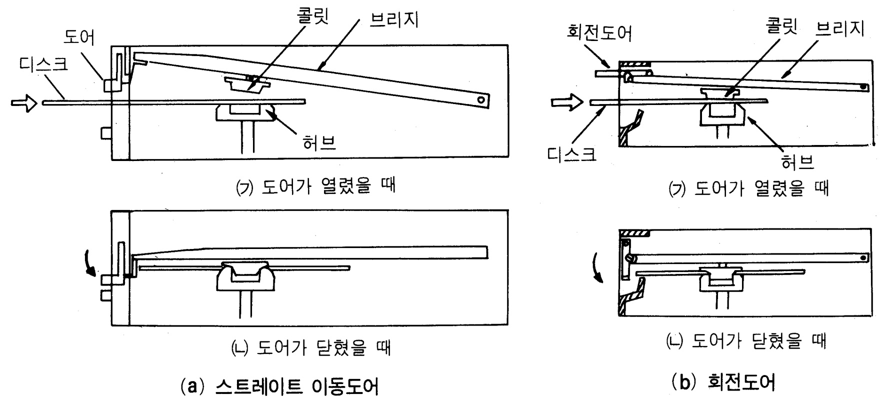
●모드 셀렉터

이것은 1.6M바이트 FDD경우에 있어서 리드/라이트하고자 하는 디스켓에 따라 리드/라이트 주파수나 특정 트랙 이상에서는 라이트 커런트(Current)를 변경시켜야 할때 동작되는 회로이다. 이때 리드/라이트 주파수에 의해 필터가 선정되어야 하며, 안쪽 트랙에서는 밀도가 증가함에 따라 라이트 커런트를 변경시킴으로써 데이터의 신뢰성을 높이는 역할을 한다.

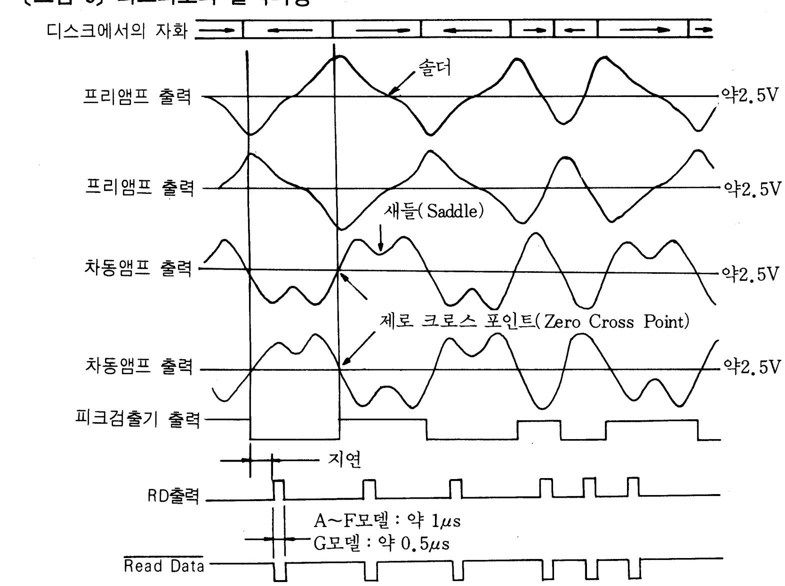
●리드회로

리드회로는 헤드 셀렉트스위치, 프리앰프(Pre-Amplifier), 저역 필터(Low Pass Filter), 차동 증폭기(Differential Amplifier), 피크검출기(Peak Detector), 타임 도메인 필터(Time Domain Filter) 및 출력 드라이버 등으로 구성되어 있다. 각 부문의 출력파형은 그림 6과 같다.

〔그림 5〕 도어 브리지 방식



〔그림 6〕 리드회로의 출력파형



●라이트회로

라이트회로는 라이트 콘트롤 로직, 라이트 커런트 소스, 라이트 드라이버, 이레이즈 드라이버 및 콤먼 드라이버 등으로 이루어져 있다. 라이트회로의 동작은 그림 7과 같다.

●로우 전압 센서

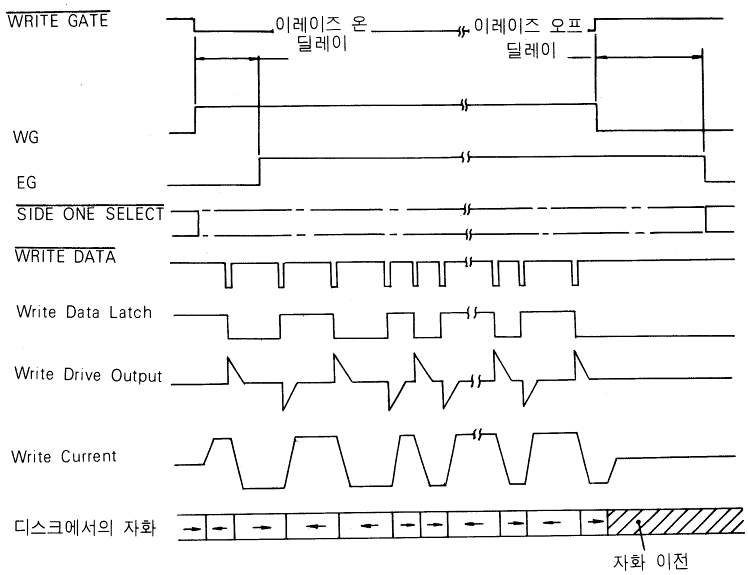
파워 ON/OFF시 불안정한 파워상태에서의 오동작을 막기 위한 회로다.

(2) 콘트롤회로

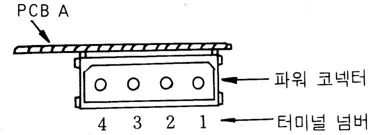
콘트롤회로는 스트랩(Strap) 회로와 헤드 로드 콘트롤회로, 라이트/이레이즈 콘트롤회로, 모터 온 게이트, 검출 준비기(Ready Detect), 스텝 모터 콘트롤회로, 트랙 카운터, 디스크 체인지 회로(Disk Change Circuit), 인터페이스 드라이버 등으로 구성되어 있다.

즉, 유저로 하여금 여러가지 기능(즉, 펑션)을 선택하게 해주는 스트랩회로, FDD를 선택해주는 드라이브 셀렉트 게이트와 헤드 로드와 관계된 헤드

〔그림 7〕 라이트회로의 동작



〔그림 8〕 파워 코넥터 구조



(a) 파워 코넥터의 핀위치

전 압	터미널 넘버
DC+12V	1
0V	2
0V	3
DC+5V	4

(b) 파워 인터페이스 코넥션

로드 콘트롤회로, 스핀들모터의 회전 조건을 결정하는 스핀들모터 게이트, 실제 기록하는 라이트/이레이즈 회로 등이 있다. 그리고 방향과 스텝 펄스에 의해 제어하는 스텝핑모터 제어회로 등이 있다.

(3) 서보회로

스핀들모터 게이트를 통해 공급된 모터 온 신호(Motor On Signal)에 의해 모터를 구동하거나 정지시키며, 0.5M바이트 FDD는 회전속도를 300rpm으로, 1.6M바이트 FDD는 360rpm 등의 듀얼 스피드 모드로 동작하게 하며 모터의 회전속도를 일정하게 유지시켜 준다.

이 스핀들모터는 3상 코일을 가진 브러시레스 DC모터이고 바이폴라 드라이브 시스템으로 코일의 에너지화 및 자기화한 방향은 모터 주위의 서보 PCB A의 홀 성분으로부터의 신호에 의해 제어된다.

이때 모터의 AC 회전속도계(Tachometer)로부터의 피드 백 신호가 전압으로 변환되며, 페이서 보상(Phase Compensation) 회로를 통해 드라이브의 코일에 흐르게 된다.

FDD 인터페이스

FDD의 인터페이스는 파워 인터페이스와 신호 인터페이스로 대별된다. 그러면 파워 인터페이스와 신호 인터페이스에 대해 알아보기로 하자.

(1) 파워 인터페이스

파워 인터페이스의 핀 수는 4핀으로 핀 위치는 **그림 8**(a)와 같다. 또한 +5V, +12V의 2전원방식으로 변경은 없으나 발열대책 등으로 소비전류는 감소하는 경향이다.

그러나 12V는 부하변동이 심하므로 주의해야 한다.

코넥터는 제품마다 통일된 규격품을 사용하므로 어느 메이커의 FDD라도 문제없이 접속된다. 그러나 소비전류는 제품마다 다르므로 주의하여야 한다. 또 미스 코넥션(Mis-Connection)을 방지하기 위하여 코넥터 하우징내에서 메카니즘적으로 해결하고 있다. 그림 8(b)에 파워 인터페이스 코넥션 부분을 나타냈다.

파워 케이블은 FDD의 최대소비전력과 코넥터의 파워전압이 고려된 케이블이어야 한다.

(2) 신호 인터페이스

표 1에서 보는 바와 같이 FDD 입력신호는 12개, 출력신호는 5개로 모두 로우액티브이며 TTL 레벨로 동작한다. 데이지 체이닝(Daisy Chaining)에 의한 멀티플렉스 코넥션으로 총 케이블 길이는 (330Ω의 터미네이터(Terminator) 일 경우) 3m이다.

코넥터에는 보통 프린트기판에 의한 34핀 카드 에지 코넥터가 사용된다. 카드 에지 코넥터는 프린트기판의 두께가 1.6㎜밖에 사용되지 않지만 각 핀 헤더는 기판의 뒷쪽에 코넥터가 나오지 않으므로 스페이스 팩커(packer)가 좋아지고 신뢰성이 높다.

그러면 표에 있는 각 핀들에 대해 알아본다. 그런데 이들이 담당하고 있는 역할이 결국 드라이브가 콘트롤러에 의해 제어되어 화일의 생성, 재생기록 등이 행해지고 있음을 알 수 있을 것이다.

●드라이브 셀렉트 0~3

데이지 체인 방식으로 최대 4대까지 연결이 가능하다. 이 신호가 전송된 후 각 신호가 액티브되는데 걸리는 시간이 인터페이스 케이블내의 지연시간을 포함해서 0.5μsec이다.

●모터 온

이 모터 온(Motor On)은 스핀들모터를 회전시키거나 정지시키는 신호로 듀얼 스피드 모드가 지원되는 드라이

브에서 하이/노말밀도 신호와 같이 사용하며 스피드를 변화시킨다.

●방향 셀렉트

스텝신호와 같이 사용되며 헤드의 움직이는 방향을 결정한다. 스텝 출력(Step Out)은 하이 레벨이고 스텝입력은 로우레벨로 정의되었다.

●스텝

헤드를 움직이게 하는 신호로써 한 펄스당 헤드를 한트랙만 움직이게 한다. 헤드 시크동작은 스텝펄스의 트레일링 에지(Trailing Edge)에서 실행되며 펄스간격은 3msec 이상이어야 한다. 이 신호에 대한 스텝 타이밍은 그림 9와 같다.

●라이트 게이트

디스크에 라이트된 데이터를 지우거나 새로운 데이터를 라이트할 때 사용되는 신호이다. 이 신호와 라이트 프로텍트신호 그리고 드라이브 셀렉트신호의 조건이 일치할 때 FDD는 라이트모드로 된다.

이 라이트 게이트 신호가 False로 된후 적어도 590μsec(360rpm)나 1msec(300rpm) 경과후에 콘트롤 신호의 레벨이 바뀌어야 한다. 이것은 이레이즈 딜레이 타임(Erase Delay Time) 때문이다.

●라이트 데이터

라이트 데이터는 라이트 게이트가 True로 됨과 동시에 라이트 프로텍트가 False가 되어야만 디스크에 쓰여질 수 있고 그림 10은 이 라이트 데이터의 타이밍을 보여준 것이다.

●사이드 원 셀렉트

디스크 양면중 어느 면을 사용할 것인가를 결정하기 위한 신호이다. 즉 하이레벨이면 Side 0의 헤드가 선택되고, 로우 레벨이면 Side 1의 헤드가 선택된다.

●트랙00

헤드가 트랙00에 위치해 있다는 것을 알리는 신호이다.

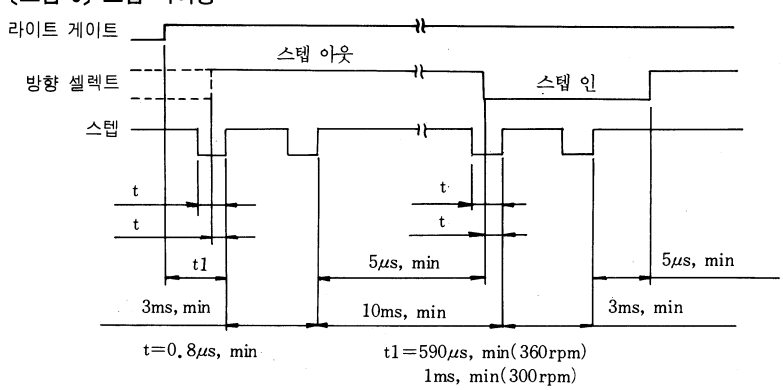
●인덱스

트랙의 시작점을 표시하는 신호이며

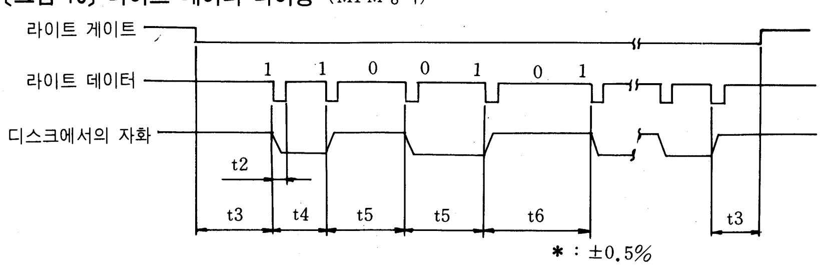
〔표 1〕 시그널 인터페이스 연결

신호	방향	터미널 넘버	
		신호	0V
하이/노말 밀도(High/Normal Density)	입력	2	1
인 유스/오픈(In Use/Open)	입력	4	3
드라이브 셀렉트 3	입력	6	5
인덱스	출력	8	7
드라이브 셀렉트 0	입력	10	9
드라이브 셀렉트 1	입력	12	11
드라이브 셀렉트 2	입력	14	13
모터 온(Motor On)	입력	16	15
방향 셀렉트(Direction Select)	입력	18	17
스텝	입력	20	19
라이트 데이터	입력	22	21
라이트 게이트	입력	24	23
트랙 00	출력	26	25
라이트 프로텍트	출력	28	27
리드 데이터	출력	30	29
사이드 원 셀렉트(Side One Select)	입력	32	31
레디/디스크 체인지(Ready/Disk Change)	출력	34	33

〔그림 9〕 스텝 타이밍



〔그림 10〕 라이트 데이터 타이밍 (MFM방식)



밀도 & 디스크 스피드	t2	t3	t4	t5	t6
High, 360rpm	0.07~1.1μs	4μs, Max.	2μs, Nom.	3μs, Nom.	4μs, Nom
Normal, 360rpm	0.07~2.1μs	6.7μs, Max.	3.3μs, Nom.	5μs, Nom.	6.7μs, Nom.
Normal, 300rpm	0.07~2.1μs	8μs, Max.	4μs, Nom.	6μs, Nom.	8μs, Nom.

디스크 1회전당 하나의 인덱스 펄스가 나온다.

그림 11은 이 신호의 타이밍을 보여준 것이다.

●리드 데이터

디스크에서 읽혀진 데이터를 콘트롤로 보내주는 인터페이스 라인이다. 그림 12는 이 신호의 타이밍도이다.

●라이트 프로텍트

디스크자켓에 있는 라이트프로텍트 노치가 마스크되어 있는지를 표시해주는 신호로써 이 신호가 True이면 디스크에 쓰여진 데이터를 지우거나 새로운 데이터를 라이트 하는 것이 금지된다.

●레디 / 디스크 체인지

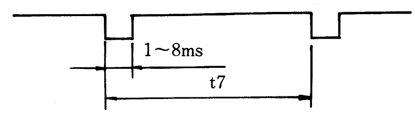
레디 / 디스크 체인지(Ready / Disk Change) 신호는 일반적으로 드라이브 PCB A에 있는 점퍼의 세팅으로 레디 혹은 디스크 체인지모드로 동작할 수 있다. 레디펑션이 선택될 때 이 신호는 FDD가 레디상태라는 것을 표시하는 것으로 파워가 ON이 되고 디스크가 세트된 상태에서 모터 온 신호가 True가 된 후 500msec가 경과되면 True가 된다.

디스크 체인지가 선택될 때는 디스크가 드라이브에 장착되어 있는가를 체크하는 것으로 파워가 ON이 되고 디스크가 제거된 경우에 이 신호는 True가 되고 디스크가 장착이 되면 False가 된다. 이것은 1.6M바이트 FDD로 0.5M바이트 디스켓이나 1.6M바이트 디스켓을 동시에 리드할 수 있어야 하기 때문에 디스켓이 교체된 후에 콘트롤러로 디스크상태를 알려주어 콘트롤러로 하여금 어떤 종류의 디스크가 드라이브에 장착되어 있는가를 결정하게 하기 위한 신호이다.

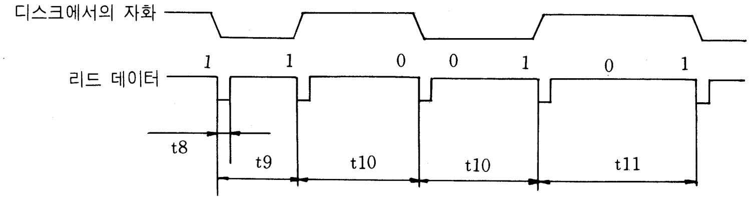
●하이 / 노말 밀도

이 하이 / 노말 밀도(High / Normal Density)에 의해 듀얼 스피드가 변화된다. 즉 360rpm은 고밀도 (하이밀도) 모드로, 300rpm은 표준 밀도(노말밀도) 모드로 변환된다.

〔그림 11〕 인덱스 타이밍



〔그림 12〕 리드 데이터 타이밍 (MFM방식)



밀도 & 디스크 스피드	t8	t9	t10	t11	t12
High, 360rpm	0.5±0.25μs	2μs, Nom.	3μs, Nom.	4μs, Nom.	±350ns
Normal, 360rpm	0.5±0.25μs	3.3μs, Nom.	5μs, Nom.	6.7μs, Nom.	±580ns
Normal, 360rpm	0.5±0.25μs	4μs, Nom.	6μs, Nom.	8μs, Nom.	±700ns

디스크와 디스크포맷

디스켓이라고도 불리우는 플로피디스크는 폴리에스터 필름(Polyester Film)의 양면에 γ-Fe_2O_3 미립자가 코팅된 것으로 사각형의 자켓으로 보호되어 있다. 그림 13에 플로피디스크에 대한 칫수를 표시했다. 라벨이 붙은 면을 Side 1, 그 반대면을 Side 0라 하며 단면디스크와 양면디스크는 완전히 같은 형상과 칫수를 갖고 있으나 디스크라벨에 양면용으로 보증하고 있느냐에 따라 달라진다.

자켓의 중심에는 디스크를 클램핑하여 회전할 수 있도록 센터홀이 있고, 자기헤드를 디스크면에 접촉해서 리드 / 라이트할 수 있도록 가늘고 긴 구멍이 있는데 이것을 헤드윈도우라고 한다. 인덱스 혹은 트랙의 스타팅 포인트를 광학적으로 검출하는 장소이며, 라이트프로텍트 노치는 디스크에 데이터를 지우거나 새로운 데이터가 라이트되는 것을 방지하기 위한 장치이다.

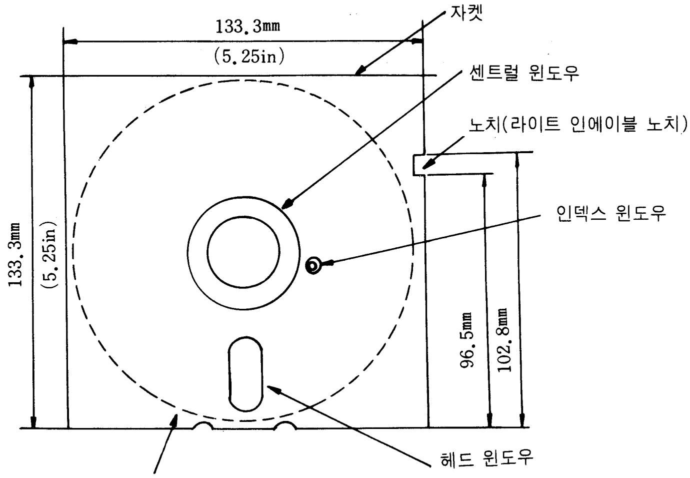
디스크상에는 맨바깥쪽에서부터 차례로 실린더 0, 1, 2, 3…이 동심원상으로 늘어서 있고 8인치 디스크에서는 실린더 76까지 5.25인치 디스크에서는 48TPI인 경우 실린더 39까지, 96TPI 인 경우 실린더 79까지이다. 또한 3.5인치 디스켓은 실린더 79까지 사용하며 1개의 트랙은 그림 14와 같이 프리앰블(Pre-amble)과 여러개의 섹터 그리고 포스트 앰블(post-amble) 등으로 구성되어져 있다.

각각의 섹터도 데이터를 리드 / 라이트 하기 편하도록 ID 필드와 데이터 필드로 나누어 구획정리를 하는데 이것을 디스크포맷이라 한다.

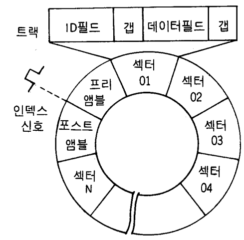
어떤 시스템에서 라이트된 디스켓을 다른 시스템을 이용하여 리드해내기 위해서는 트랙의 위치 즉 인덱스 홀의 상대적위치가 같아야 하며, 또한 디스크포맷도 같아야 한다. 만약 디스크포맷이 서로 호환성이 없다면 드라이브는 디스켓을 리드할 수 없게 된다.

트랙을 라이트하기 시작하는 위치는 FDD의 인덱스 센서가 디스크의 인덱스홀을 검출하는 곳이 되는데, 이 인덱스의 센서와 헤드는 각 FDD마다 약간씩 위치가 어긋나 있다. 때문에 다른

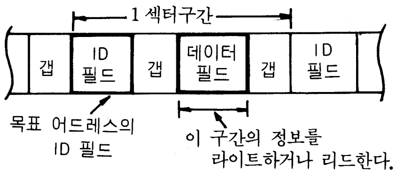
〔그림 13〕 플로피디스크의 외관



〔그림 14〕 디스켓 트랙의 구조



(a) 디스크 및 디스크 포맷



(b) 섹터

FDD에서 리드할 경우 트랙의 스타트 위치가 인덱스신호보다 빠를 수 있다.

따라서 그 시간차만큼 디스크포맷에 공간이 필요하게 된다. 이것이 프리앰블(Pre-amble)이며, 각 섹터마다 섹터의 시작위치와 동기필드(Sync Field), 그리고 섹터어드레스를 표시하는 정보를 기록한 ID필드와 실제의 데이터를 기록하는 데이터 필드가 있다.

이들 전후에는 회전속도 변동이나 기계적 칫수오차, 이레이즈 게이트의 지연이나 타이밍 오차로부터 데이터를 보호하기 위해 갭이 놓여져 있다. 트랙 한바퀴에서 프리 앰블과, 여러가지 섹터를 배열한 나머지를 디스크 회전변동 등으로 인해 발생할 수 있는 데이터 에러를 가이드(guard)하는 곳이 포스트 앰블(Post-Amble)이다. 이와 같은 사실때문에 포스트 앰블의 바이트 수는 1회전 속도 변동분보다 많아야 한다.

따라서 이와같은 사항을 고려하여, 1트랙당 섹터수를 16, 9, 5 등으로 나눌 수 있으며, 1섹터당 바이트수를 최적인 것으로 선정한다. 즉 1트랙당 섹터수가 작으면 갭이 차지하는 비율이 감소하기 때문에 디스크 용량은 많아진다.

한편 섹터를 구분하는 방식은 하드 섹터방식과 소프트 섹터방식이 있다. 전자는 디스켓 자체에 섹터 홀이 뚫어져 있어서 이것으로부터 발생되는 신호를 이용 섹터를 구별하는 것이며, 후자는 1개의 인덱스 홀을 통해 기준 섹터를 결정한 다음 다른 섹터는 소프트적으로 정해서 쓴다(그림 15 참조).

기록방식

실제 디스켓에 기록하는 방식으로는 FM, MFM 외에 M^2FM, GCR, RLL 등이 있는데 여기서는 FM방식과 현재 표준으로 되어 있는 MFM 기록방식에 대해 알아보자.

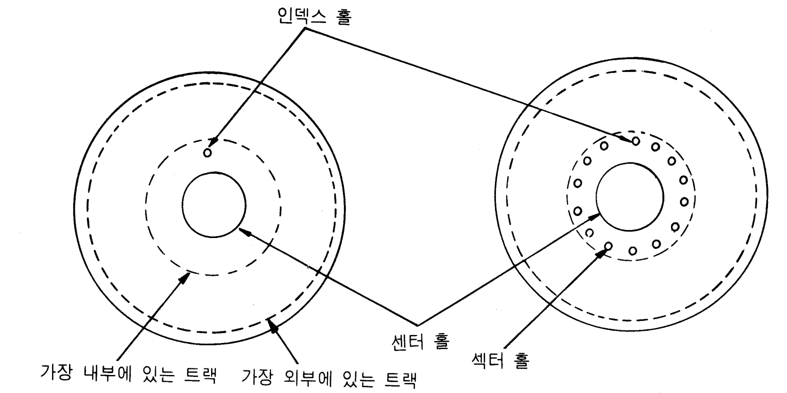
FM방식이란 그림 16과 같이 비트셀의 선두에 클럭비트를 라이트하고 비트 셀의 중앙에 데이터비트를 라이트하는 방식으로 데이터 1비트의 간격이 8μs이다. 결과적으로 기록신호는 데이터가 "1"일 때 4μs이고, "0"일 때는 8μs간격이 되어 주파수변조(Frequency Modulation)로 매체에 기록된다.

반면 MFM방식은 FM방식의 기록에서는 매 비트셀마다 항상 클럭비트를 쓰는데 반해서 데이터가 없는 경우에만 클럭비트를 삽입하는 방법으로 같은 플럭스(Flux) 밀도로써 FM에 비해 기록밀도를 2배로 실현시킬 수 있다.

즉 FM방식과 같이 4μs까지 단축하여 기록밀도를 2배로 해도 매체에서 본 기록 주파수대역(Frequency Band) 즉 FCPI(Flux Change Per Inch)와 같은 셈이 되고 FM방식의 2배 기록밀도를 실현하는 것이다. 이러한 이유로 MFM 기록방식을 배밀도(Double Density) 기록방식이라 한다.

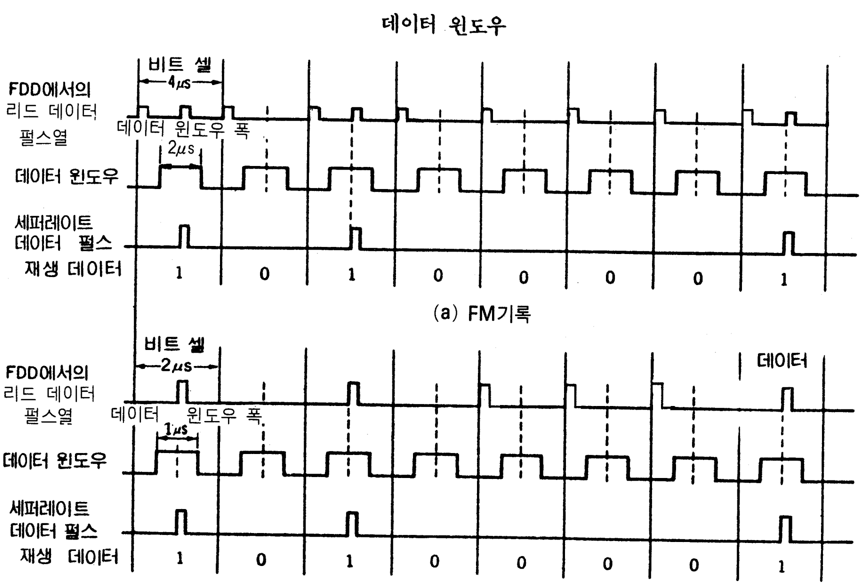
그림 17과 같이 서로의 독립파형이 매체에서 리드될 때 서로의 독립파형이 간섭을 일으키며 합성파형은 피크부분의 위치가 어긋난다. 이것을 피크 시프트(Peak Shift)라 한다. 5.25인치인 경우 펄스간격이 4μs, 6μs, 8μs로 이루어져 있으므로 리드 데이터에서 클럭과 데이터를 분리하는 데이터 윈도우는 이 피크 시프트에 대해 충분한 간격을 가져야 한다. 이 경우 데이터 윈도우 폭은 2μs로 안쪽 트랙으로 갈수록 기록밀도가 높아지므로 피크 시프트양도 커진다. 이 때문에 피크 시프트양이 적어지도록 헤드를 포함하는 리드 채널에 대한 설계가 필요하며 라이트 커런트 등을 조정하여 안쪽트랙

〔그림 15〕 소프트 섹터와 하드 섹터방식



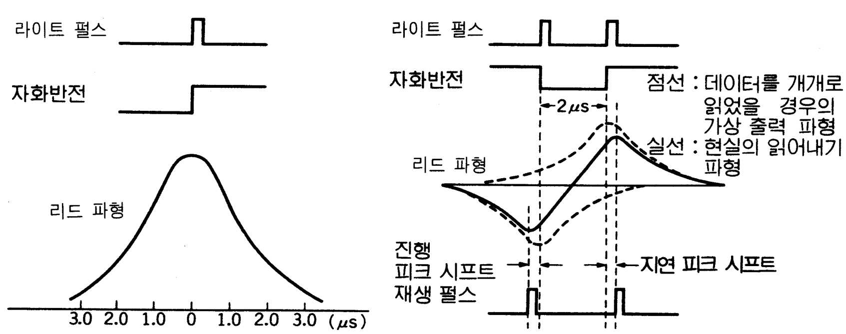
(a) 소프트 섹터된 디스크　　(b) 하드 섹터된 디스크(16섹터)

〔그림 16〕 FM, MFM기록방식의 비교



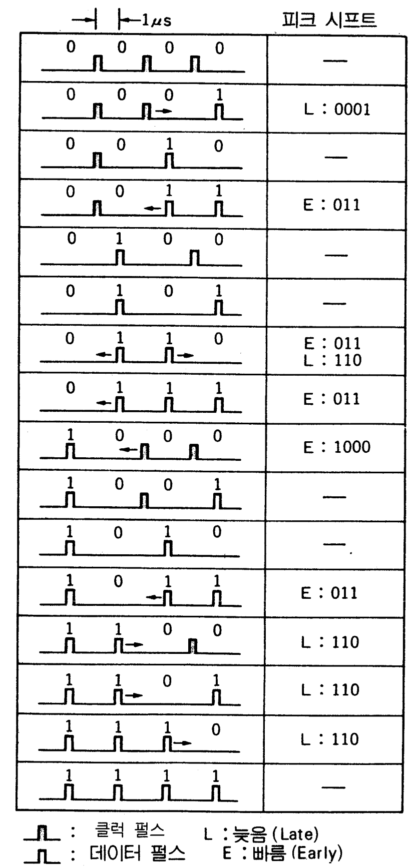
(b) MFM기록

〔그림 17〕 파형의 간섭



(a) 단독파형　　(b) 간섭에 의한 피크 시프트

〔그림 18〕



(a) 각 데이터 패턴에서의 피크 시프트

피크 시프트 늦음(L)	피크 시프트 빠름(E)
0001	011
110	1000

(b) 피크 시프트가 발생하는 데이터 패턴

에서의 분해능(Resolution)을 높인다.

또한 피크 시프트의 방향은 데이터의 패턴에 의해 예상할 수 있기 때문에 미리 라이트할 때 피크 시프트방향과 반대방향으로 움직여 놓음으로써 그 영향을 감소시킬 수 있다. 이것을 라이트 프리보상(Write Pre-Compensation) 이라 한다.

그림 18(a)는 각 데이터 패턴에 따라 리드할 때의 피크 시프트 방향을 표시한 것이다. 그림 18(b)와 같은 데이터패턴에서 피크 시프트가 발생하므로 데이터를 디스크에 라이트하기 전

에 이와 같은 패턴을 검출해서 기록속도를 조절할 수 있다. 이것은 플로피디스크 콘트롤러에서 보상을 하여 드라이브로 데이터를 전송한다.

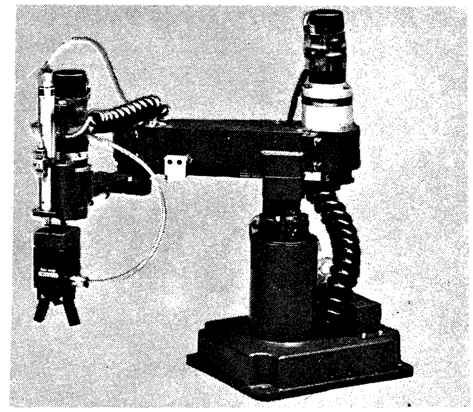
맺음말

FDD의 제품 및 개발추세는 HDD와 마찬가지로 3.5인치가 시장을 주도해 나갈 것이며 용량도 0.5M→1.6M→2MB로 옮겨가는 중이다. 특히 랩탑 컴퓨터용으로 로우 프로파일(Low Profile) 3.5인치 FDD가 발표되고 있으며 2.5인치 등의 소형 FDD의 개발 등이 이루어지고 있다.

수년전 IBM-PC가 발표되면서 XT인 경우 0.5MB의 5.25인치를 AT인 경우 1.6MB의 5.25인치 FDD를 장착하였다. 그리고 작년에 IBM이 PS2를 발표하면서 1MB, 2MB의 3.5인치 FDD을 채택하였다.

즉 현재 5.25인치 FDD에서 3.5인치 FDD로 시장이 변하고 있으며 시스템의 발전추세에 따라 그에 따른 발전속도와 신소재 및 새로운 기술개발 등의 변화를 계속적으로 추구해 나갈 것이다. PC

마이크로프로세서를 이용한
FA SYSTEM과정
(생산자동화)



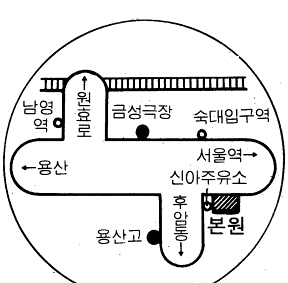
※ 개강 : 매월 5일

講義案內 (2개월 완성)

본원에서는 다년간 Computer Hardware 과정을 교육하여 많은 수료생을 배출, 각계에서 공헌하고 있는 바 금번 생산업계의 요청에 따라 컴퓨터 프로세서를 이용한 생산체제와 산업용 로보트의 발전으로 제품과 제조공정의 설계 및 계획, 공장 현장에 있는 생산 제조장치의 컴퓨터 제어, 원료 및 자재의 유동제어, 공간 활용 및 작업계획 등의 조직적 기획 등의 필요성에 따른 FA 과정의 이론, 실기를 위주로 특강을 하게 되었으니 관계 직원의 많은 참여 바랍니다.

■ **강의일정**

주별	일자 \ 과정	강습내용 이론및실기	주별	일자 \ 과정	강습내용 이론및실기
1주	1일차	Logic회로 설계의 기초	5주	1일차	PWM System
	2일차	Micom System		2일차	Power Control System
	3일차	KKH 80A H/W구조		3일차	DC Motor Control
2주	1일차	Z80 Assembly 언어	6주	1일차	Step Motor의 기초 이론
	2일차	KKH 80A S/W 구조		2일차	Step Motor Control
	3일차	PIO 입출력 실습		3일차	Senser Interfacing(1)
3주	1일차	A/D Converter의 제작	7주	1일차	IBM PC H/W 구조
	2일차	D/A Converter의 제작		2일차	IBM PC/XT Interfacing
	3일차	DC Motor Drive회로의 설계		3일차	RS 232C Serial Interfacing
4주	1일차	Monitor Program의 설계	8주	1일차	Senser Interfacing(2)
	2일차	응용 Program 설계(1)		2일차	Micro Robot(1)
	3일차	응용 Program 설계(2)		3일차	Micro Robot(2)



전자산업 4 반세기를 지켜온 기술인의 전당

과인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

스피디 3의 제작

전윤호/서울대 제어계측공학과 대학원

마이크로 로보트 경연대회는 횟수를 거듭할 수록 그 인기도를 더해가고 있다. 이번호부터 2회에 걸쳐 지난 10월29일 열린 제6회 마이크로 로보트 경연대회에서 최우수상을 탄 '스피디 3'에 대해 설계상의 노하우와 제작과정 등을 알아보고자 한다.

마이크로 로보트 경연대회는, 이미 많이 알려져서 구태여 설명할 필요는 없겠으나, 각 참가자가 만든 소형의 로보트가 정해진 규격을 가진 미로에서 스스로 얼마나 빠르게 길을 찾아가는지를 겨루는 기록 경기이다.

필자는 이 대회에 여러번 참가했었는데, 1987년 제5회 마이크로 로보트 대회에서 Speedy-2가 우승을 하고 난 후, 미국 대회에 대비해서 Speedy-2를 대폭 개조하고자 했었다. 그 시도가 결국 실패했었는데, 그 이유로는 시간의 부족과 함께 Speedy-2의 기본적인 구조자체가 속도에 여러가지로 제약을 주었기 때문이다. 그 자세한 내용은 앞으로 설명하겠으나, 어쨌든 그 이후로 제6회 대회에 대비하여 더욱 우수한 성능의 마이크로 로보트를 개발하기로 했다.

새로운 마이크로 로보트의 이름은 당연히 Speedy-3으로 결정하였고, 애초에 목표로 했던 것은 아래와 같은 것들 이었다.

- 스텝모터의 정전류 구동에 의한 고속화
- 곡선회전 (소위 Smooth Turn)의 실현
- Loop Test의 적용
- 최단시간 경로의 계산

위의 목표들을 위해서는 하드웨어 면에서 아래와 같은 것들이 요구되었다.

- 더욱 빠르고 복잡해진 여러 처리를 위한 16비트 CPU
- 대용량의 메모리
- 오차 최소화를 위한 많은 수의 센서

이상 많은 계획이 구상되었으나 실제로 대회전에 구현할 수 있었던 것은 그들 중 일부분에 지나지 않았다. 그렇지만 개발과정에서 겪은 많은 경험이 앞으로 고성능의 마이크로 로보트를 만들어 대회에 참가하고자 하는 사람들에게 도움이 될 것으로 생각하여 Speedy-3에 적용된 모든 것과 함께 그 동안 고려되었던 많은 구상, 기술 등을 모두 공개하고자 한다.

마이크로 로보트의 특성상 설명의 내용이 어렵고 이에 대한 상당한 사전 지식이 없는 사람들에게는 이해하기가 곤란한 부분이 있음에 대해 사전에 양해를 구하며 경기 규정이나 보다 기초적인 내용에 대해서는 다른 문헌을 참조하기 바란다. 또, Speedy-3에 사용된 기술중 많은 것들이 Speedy-2에 사용되었던 기술과 같거나 그 연장이다. 따라서 앞으로의 설명중 어떤 부분은 Speedy-2에 관한 설명 (월간 제어계측기술 88년 1, 2월호)과 큰 차이가 없음을 양해하기 바란다. 설명 중에는 기술적인 우수성에도 불구하고 지난 대회에서 아깝게 준우수상을 받은 Turbo-4에 대한 언급이 여러번 나오는데 이에 대한 자세한 것은 월간 마이크로소프트웨어 88년 12월, 89년 1월호를 참조하기 바란다.

전체의 구성

마이크로 로보트는 매우 작은 시스템이지만 여러가지 제약 조건이 많기 때문에 한번만에 최적의 설계를 하기란 대단히 어렵다. 각 부분에 사용할 회로는 요구되는 결과로서의 성능 뿐만 아니라 회로가 차지하는 부피나 무게, 소비 전력, 전원 전압 등에 의해 적합성이 좌우되기 때문에 그 설계가 어렵다. 특히 한 부분의 변경은 밀접하게 연결된 다른 부분의 설계에도 변경을 요구하게 되는 경우가 많아서 기능단위별로 독립적인 설계를 할 수가 없다. 예를 들어 모터 드라이브 회로의 종류에 따라 전체 전원이 결정되고, 이에 따라 CPU나 센서 등에 공급해야 하는 전원을 어떻게 마련해야 하는지의 방법이 달라진다. 특히 전체 회로를 제한된 공간에 집어넣어야 하기 때문에 회로를 각 기판에 어떻게 분배하고 그 사이의 연결은 어떻게 할 것인지도 쉬운 문제가 아니다.

초보자들의 경우에는 모든 부분에 어느 정도의 여유를 두는 것이 중요하

다. 나중에 회로가 일부 추가되거나 변경되는 경우가 매우 많기 때문이다. 설계의 초기 단계는 어떤 정해진 원칙이 있다기 보다는 설계자의 경험과 창의에 많이 의존하는 것으로서 여기에 많은 노력을 투자해야 나머지 부분이 쉬워지고 좋은 성과를 거둘 수 있게 된다.

고속의 마이크로 로보트를 제작하려면 전자회로만이 중요한 것은 아니다. 경주용 자동차를 설계하듯이 하라면 과장이겠으나 어쨌든 30초 이내의 기록을 내기 위해서는 무게중심의 높이, 바퀴의 위치같은 문제를 간과할 수가 없다. Speedy-2는 무게중심이 비교적 높았기 때문에 급격한 가/감속시에 비퀴가 쉽게 미끄러졌다. 무게중심을 낮추기 위해 무거운 부품을 아래로 배치하다보면 미로벽에 닿기 쉽고, 또 중심에서 떨어진 곳에도 배치를 해야 하므로 회전관성이 커져서 회전속도가 느려진다. 물론 각 센서를 어디에 설치하는가 하는 것이나 앞/뒤의 보조 바퀴의 종류도 설계에 영향을 준다.

이외에 여러 요소가 전체 구성과 모양에 영향을 주는데 이중 많은 것들-예를 들면 각 보드의 크기 등 -은 제작 도중에나 알 수 있다. 따라서 실제의 초기 단계에 최적의 설계를 할 수는 없고 예상되는 정도에서 약간의 여유를 두고 설계한 후, 다른 부분을 설계하고, 만일 안맞으면 먼저 부분을 다시 설계하고 하는 식으로 여러 부분을 반복하여 설계해서 맞춰보는 수 밖에 없다. 그러므로 Speedy-3가 앞으로 여기서 설명하는 순서대로 단계적으로 설계된 것은 결코 아니고 계속된 시행착오를 거쳐 지금의 형태가 되었음을 미리 밝혀주고자 하며, 그 시행착오의 과정을 일일이 나열하기에는 지면이 부족하므로 완성된 형태만을 위주로 하여 설명하는 것에 대해 양해를 구한다.

Speedy-3의 전체 형태를 결정짓기 위해서는 최소한 모터와 센서가 우선적으로 결정되어야 했다. 앞으로 설명할 여러 이유 때문에 스텝모터 2개와 32조의 적외선 센서, 1조의 초음파 센서를 사용하기로 결정했고 센서가 부착되어야 할 위치나 무게중심, 필요한 여러 회로기판의 추정면적과 적절한 배치위치 등을 고려하여 전체적인 모습을 사진 1과 같은 형태로 결정했다.

이는 상당한 모험이었는데 기판을 베치하는데 있어 여유가 거의 없었기 때문이었다. 결국 각 부품을 최대한 밀집시켜 배치하고, 가능하면 회로를 더욱 간단한 것으로 바꿔 문제를 해결했다. 완성된 모습은 사방에 노출된 회로기판이 붙어있는 흉칙스러운 것이 되었으나, 성능을 위주로 하다보니 어쩔 수 없는 일이었다. 각 기판의 배치는 그 기능이나 필요 면적에 따라 결정되었는데 윗면에 CPU, 앞면과 날개상에 센서 및 그 처리회로, 뒷면에 모터 드라이브 회로가 부착되었다.

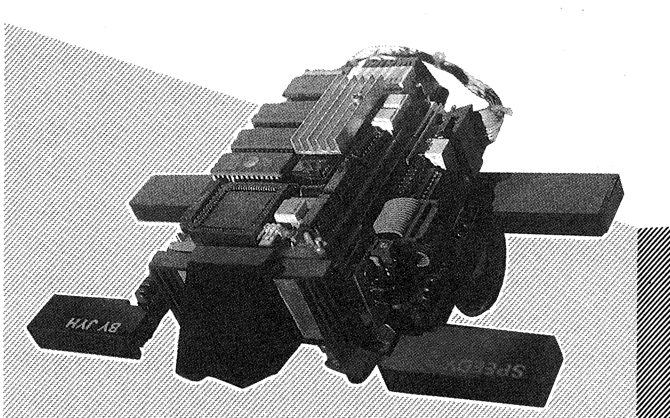
모터

마이크로 로보트를 개발함에 있어 가장 기본적인 선택은 구동장치이다. 현실적으로는 바퀴를 사용해서 굴러가는 수 밖에 없은데, 바퀴를 어떻게 배치하며, 회전을 어떻게 할 것인가가 문제가 된다. 일반적으로는 두개의 모터를 사용해서 좌/우 바퀴를 각각 구동하는 것이다. 이렇게하면 좌/우의 바퀴를 서로 반대 방향으로 돌림으로써 제자리 회전이 쉽게 된다.

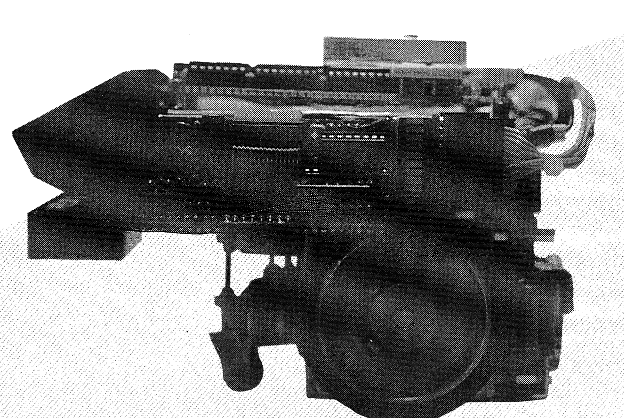
물론 보통의 자동차처럼 만드는 것도 생각할 수 있지만, 좁은 미로라든지 자동장치의 복잡함 등을 생각하면 개인적으로 만들기는 어렵다는 것을 알 수 있다.

그 다음은 어떤 종류의 모터를 선택하느냐는 것이다. 가장 일반적으로 쓰이고 있는 것이 DC 서보모터(Servo Motor)와 스텝모터(Step Motor)이며 이들 모터를 마이크로 로보트에 적용할 경우의 장단점은 각각 다음과 같다.

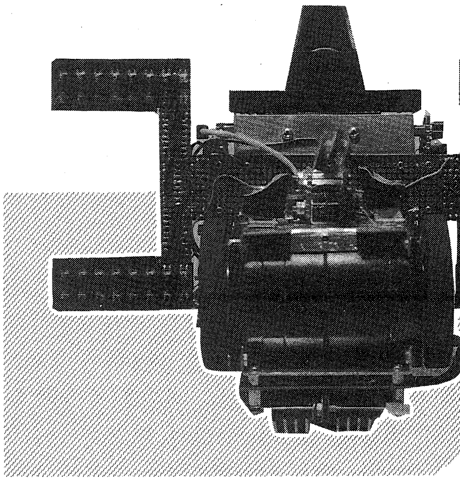
〔사진 1〕 SPEED-3의 외관



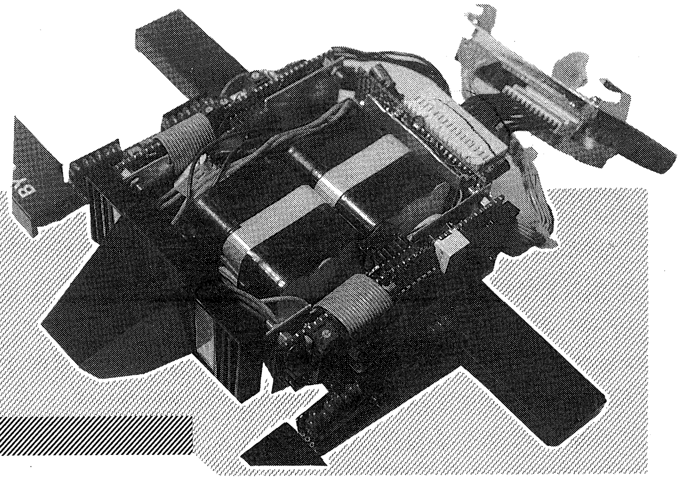
〔사진 2〕 옆에서 본 모습



〔사진 3〕 밑에서 본 모습



〔사진 4〕 CPU 보드를 제거하면 축전지가 나타난다



DC 서보모터의 장점

–소형, 경량이다. 따라서 마이크로 로보트 전체를 작고 가볍게 만들 수 있다. 특히, 폭이 좁아지면 측면 벽에 충돌할 가능성이 줄어들고 곡선회전에 유리하다.

–전력 소모가 적다. 따라서 작은 전지를 쓸 수가 있다.

–고속 회전에 특히 유리하다.

–제어 회로가 비교적 덜 복잡하다.

DC 서보모터의 단점

–정밀한 로터리 인코더(Rotary Encoder)와 소형의 감속 기어가 필요하다.

–정확한 제어가 쉽지 않고, 제어를 잘 하기 위해서는 복잡한 이론과 함께 주행시 많은 계산이 필요하다.

스텝 모터의 장점

–제어가 쉽다. 단지 펄스를 주기만 하면 된다.

–바퀴의 미끄러짐을 제외하면 오차가 없다.

–Open–loop 제어가 가능하고 계산량이 적다.

–바퀴에 직접 연결할 수 있으므로 제작이 쉬워진다.

스텝 모터의 단점

–크고 무겁다.

–고속 회전이 어렵다. 고속 회전을 위해서는 높은 전압과 복잡한 회로가 필요해진다.

–탈조를 막기 위해서는 토크에 여유가 있어야 하므로 성능을 다 활용할 수 없다.

–전력 소모가 매우 많다. 따라서 전지 무게도 무거워진다.

위에서 알 수 있듯이 두 모터는 서로 반대되는 장단점을 지니고 있다. 일반적으로 처음 마이크로 로보트를 만드는 사람은 스텝 모터를 사용하고, 우수한 성능을 얻기 위해서는 DC 모터를 사용하는 것으로 알려져 있는데 Speedy –3에서는 Speedy–2와 마찬가지로 계속해서 스텝모터를 사용하기로 했다.

그 이유는 DC 서보모터를 사용하여 Speedy–3를 개발하기 위해 필요한 인력과 시간이 너무 큰탓도 있으나 실험에 의해 스텝모터도 정전류 구동을 하면 마이크로 로보트에 충분한(?) 정도의 성능을 얻을 수 있음을 알았기 때문이다. 실제로 완성된 Speedy–3의 최고직진 속도는 1.5m / sec 정도로서 대회 미로의 크기를 생각하면 충분히 빠른 값이고, 그보다도 기록에 더 많은 영향을 주는 초기 가속도는 바퀴가 미끄러지지 않을 범위로 한정되기 때문에 DC 서보모터라고 해서 훨씬 더 유리한 것은 아니다.

물론, 요즘 대회의 기록이 계속 좋아지고 있는 것을 고려하면 궁극적인 속도 향상을 위해서는 DC 서보모터를 사용할 수 밖에 없을 것이나, 그 보다는 다른 측면에서 속도를 올리는 것이 더 쉬울 것이다. 소형 DC 서보모터를 다루어본 경험이 있는 사람이라면 도전해 볼 만 하겠으나, 처음 만드는 사람이라면 절대적으로 스텝모터를 사용할 것을 권한다.

일단 스텝모터를 사용하기로 했더라도 여러 스텝모터 중 어떤 것을 선택할지가 문제가 된다. 현재 우리나라에서 직접 구할 수 있는 스텝모터도 그 종류가 꽤 많지만, 그중에서 마이크로 로보트에 사용하기에 적합한 크기, 무게, 토크, 전력소모 등을 갖춘 것은 그리 많지 않고, 필자가 사용을 고려했던 것은 일본 오리엔탈사의 PH264–01A와 PXB43H–01A였다. 잠시 후에 다시 설명하겠으나, 이들 모터는 같은 계열중에서 각각 가장 낮은 4볼트의 정격전압을 갖는데, 정전류구동을 위해서는 정격전압이 낮을 수록 유리하다.

두 모터의 특성은 표 1을 참고하기

〔표 1〕 스텝모터의 규격

품　　명	전압〔V〕	전류〔A〕	여자 최대 정지 토크〔kg·cm〕	권선저항〔Ω/상〕	권선인덕턴스〔mH/상〕	모터 관성 모멘트〔g·cm²〕	중량〔kg〕
PH264-01	4	1.1	2.9	3.6	3.6	57	0.4
PXB43H-01A	4	0.95	1.1	4.1	1.6	16	0.2

바란다. PH264는 Speedy-2에 사용했었기 때문에 그 성능을 대충 알고 있었는데, PXB43H를 시험해 본 결과, 모터의 제원에서도 알 수 있는 바와 마찬가지로 고속 특성은 상대적으로 우수했으나 마이크로 로보트에서 사용하는 범위안에서는 큰 차이가 없었고, 전력소모도 비슷한 데 비해 토크는 1/2이 채 안되었다. 무게는 PH264의 1/2이고 크기도 상당히 작으므로 마이크로 로보트 전체를 작고 가볍게 만드는데 유리하지만, 전체 무게도 1/2이하로 줄이지 않으면 초기 가속도가 PH264를 사용하는 것보다 느려지게 된다. 초기 가속도는 기록시간에 가장 큰 영향을 미치는 중요한 요소중의 하나이고 Speedy-3의 무게를 Speedy-2의 1/2 이하로 줄이는 것은 불가능할 것으로 판단되었기에 결국 PH264를 사용하기로 결정했다.

참고로, Turbo-4는 PXB43H를 사용하고 여러가지 기술을 사용하여 무게를 대폭 줄임으로써 우수한 성능을 발휘했고 작으 크기 덕분에 곡선 회전을 구현하기가 쉬웠기에 좋은 성적을 낼 수 있었다. 그러나 Turbo-4는 무게를 줄이느라 전지용량에 여유를 두지 못해서 미로탐색 시간이 길이지면 센서와 모터의 동작이 불안정해진다는 문제가 있었다.

스텝모터를 사용하기 위해서는 그 구동 방식을 알 필요가 있다. 여기서 사용하는 PH264-01A는 2상 스텝모터로서 A와 B의 두 코일을 가지며 이 두 코일에 서로 90°의 위상차를 갖는 전압을 가함으로써 회전하게 된다.

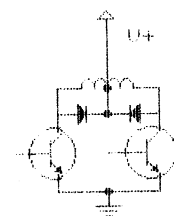
회전 방향은 어느 상의 파형이 앞서느냐에 따라 결정되며 회전속도는 입력된 전압의　상이 바뀌는 속도에 그대로 따르게 된다.

구체적인 구동방식은 여러가지로 나뉘는데, 우선 코일에 일정한 전압을 인가하는지, 또는 일정한 전류를 흘리는지에 따라 정전압/정전류 구동으로 구분되고, 각 코일에 반대 방향의 자력을 발생시키기 위한 방법에 의해 유니폴러/바이폴러(Unipolar/Bipolar) 구동으로 구분된다. 스텝모터를 구동하는데 있어서 가장 주의해야 하는 것은 탈조이다.

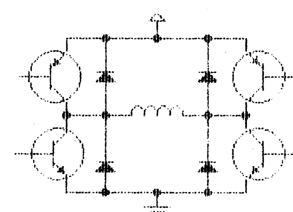
탈조란 입력주파수의 변화가 너무 빠르거나 부하 토크가 커서 모터가 입력되는　상을 따라 돌지 못하는 경우를 말한다. 운이 좋으면 한두 스텝을 놓치고 말겠지만 운이 나쁘면 모터가 완전히 정지해 버린다. 특히 더욱 문제가 되는 것은 대부분의 스텝모터가 오픈 루프에 제어된다는 것이다. 따라서 모터가 정지하더라도 CPU에서는 그 사실을 알 수가 없다. 이를 피하는 가장 확실한 방법은 모터를 여유있게 사용하는 것이다. 특히 마이크로 로보트의 경우 가속중에 진행방향의 오차를 발견하면 한쪽 모터의 속도를 갑자기 바꿔서 진로를 변경하게 되는데 이때 토크에 여유가 없으면 탈조해버리게 되며 그 모터는 완전히 정지해 버리기 때문에 옆으로 돌면서 벽에 부딪히게 된다. 스텝모터의 토크 특성을 보면 어떤 구동방식을 사용하든지 고속에서는 토크가 줄어들게 되므로, 고속이 될수록 가속률을 낮춰야 하는 이유가 여기에 있다.

스텝모터에서 좋은 결과를 얻기 위해서는 정전류 구동이 필수적이다. 그에 대한 자세한 내용은 스텝 모터에

〔그림 1〕 스텝모터의 구동회로



(a) 유니폴러의 경우



(b) 바이폴러의 경우

관한 서적을 참고하면 알 수 있겠지만, 간단히 말하자면, 고속에서 필요한 토크를 얻기 위해서는 각 상에 흐르는 전류의 방향이 빨리 바뀌어야 하는데 코일의 인덕턴스때문에 전류의 변화가 느리게 나타난다. 따라서 정격보다 더 높은 전압을 인가하여 전류의 상승속도를 빠르게 하면서 동시에 전력소모를 정격 이하로 하기위해 정전류 회로에 의해 최대 전류를 제한해야 하는 것이다.

한편 저속에서의 토크를 개선하고 에너지 효율을 높이기 위해서는 바이폴러 방식의 구동을 해야 유리하다. 그림 1의 두 회로를 비교해 보면, 유니폴러의 경우에는 한번에 전체 코일의 1/2만 사용하는데 비해 바이폴러의 경우에는 항상 전체 코일이 모두 사용되고 있음을 알 수 있다. 또, 상이 바뀔 때 코일에　남아 있던 에너지가 어떻게 되는가를 보면 유니폴러의 경우에

는 다이오드와 저항에서 열로 소모되어 버리지만 바이폴러에서는 전원으로 되돌려진다는 것을 알 수 있을 것이다. 실제로 실험을 해보면, 바이폴러의 경우에는 회전속도가 빨라지면 전류소모가 급격히 줄어들기 때문에 유니폴러에 비해 훨씬 유리하다는 것을 알 수 있다. 또 유니폴러의 경우에는 전류 상승이 정전류 회로에 의해 빠르게 이루어지더라도 전류의 하강은 L / R의 시상수에 의해 결정되어 비교적 느리지만 바이폴러 경우에는 반대방향의 전압이 인가되므로 빠르게 상을 바꿀 수 있다.

바이폴러의 나쁜 점이라면 회로가 상대적으로 복잡해진다는 것이다. 각 상마다 파워 트랜지스터가 4개씩 필요하므로 2상 모터를 2개 사용했을 때 파워 트랜지스터가 16개 필요해지고 거기에 파워 트랜지스터의 베이스 전류를 공급해주는 드라이브 트랜지스터까지 고려하면 상당히 복잡한 회로가 된다.

Speedy-3에서는 회로를 될 수 있는데로 간단히 하고 베이스 전류에 의한 불필요한 전력소모를 막기위해 파워 MOS FET를 사용하기도 했다. MOS FET는 전압에 의해 구동되므로 ON/OFF에 필요한 에너지가 거의 없고, 또 순방향 전압 강하도 줄일 수 있으므로 유리하지만 바이폴러 회로에 적용하기 위해서는 두 가지의 문제점이 있다. 먼저 바이폴러 회로에서는 p-채널의 MOS FET가 있어야 하는데 파워 MOS FET는 거의가 n-채널로서 p-채널은 구하기가 어렵다는 점이 있다. 다행히도 쓸만한 p-채널 파워 MOS FET를 구할 수 있었는데 트랜지스터를 사용한 경우에 비해 상당히 비용이 더 들었다.

또 바이폴러방식에서 정전류회로를 구성하기 위해서는 구동 트랜지스터의 콜렉터와 에미터간, MOS FET에서는 드레인과 소스간에 보호 다이오드가 없어야 전류에 의한 피드백(Feedback)이 가능해 지는데 파워 MOS FET의 경우에는 이 다이오드가 구조적으로 내장되어 있기 때문에 전류의 검출이 어려워진다. 그래서 결국 전류검출에 의한 정전류회로는 포기하기로 하고 회전속도에 따라 전원에 직렬로 저항을 넣었다 뺐다 하는 Bi-level 방식의 구동을 하기로 최종 결정했다.

이는 물론 몇번의 시행착오를 거쳐서 얻은 결과인데 완성된 보드는 추가된 회로에 의해 상당히 복잡하면서도 사용하지 않는 회로를 일부 포함하고 있는 등, 비효율적인 면이 있었다(그림 2, 사진 5 참조).

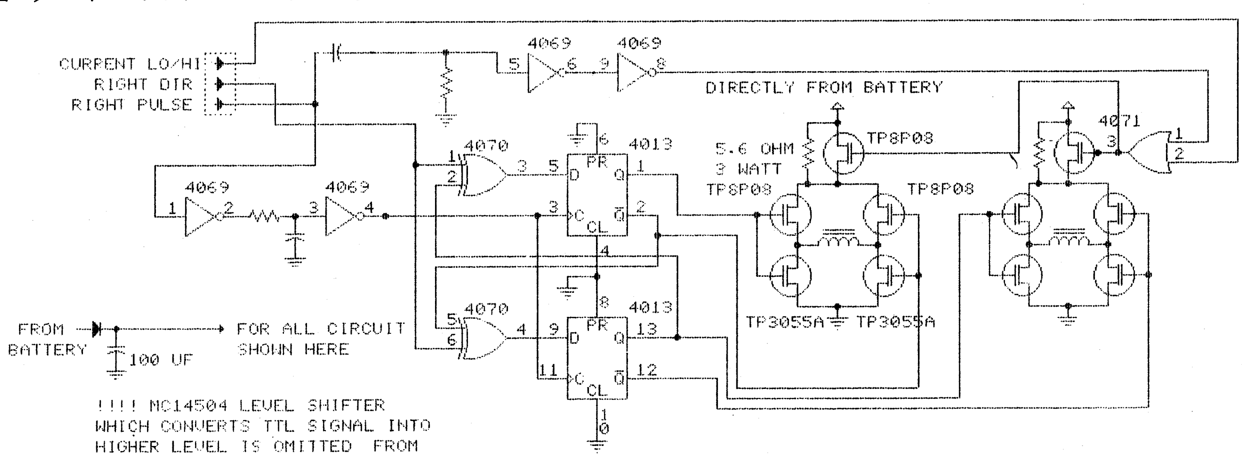
바이폴러 회로의 경우 주의할 점은 역방향의 전압을 인가하기전에 약간의 딜레이가 있어야 한다는 점이다. 만일 그렇지 않다면 스위칭 소자의 지연때문에 위 / 아래가 모두 on되어 전원이 단락되는 순간이 생길 수 있다. 이는 전원인 축전지에 좋지않고, 또 전원전압의 순간적인 변화를 일으켜서 다른 회로가 오동작하는 원인이 될 수 있다. 무엇보다도 파워 트랜지스터나 MOS FET 등이 파괴되기가 쉽다. 이를 방지하기 위해 R / C에 의한 지연회로를 사용하는데, 파워 트랜지스터의 경우 수십 μs, MOS FET의 경우 수 μs 정도면 충분하다.

아뭏튼 어렵게 완성을 했는데 15V 전원으로 무부하 실험에서 10K PPS (Pulse Per Second) 넘게 회전이 되는 좋은 결과를 보였다. 고속에서의 전류소모는 수백 mA 이하의 적은 값을 나타냈는데 부하가 있으면 전류 소모가 커지는 바람직한 현상이 보였으나 그 이유는 알 수 없었다.

CPU의 선택

원래 Speedy-2에서는 4MHz Z-80A를 사용하고 있었고, 이는 대부분의 마

〔그림 2〕 모터 드라이버 보드의 회로 구성(오른쪽 부분만)



이크로 로보트와 같은 것이다. 그러나 속도를 대폭 올리기로 생각하고 있었기 때문에 스텝모터의 인터럽트가 더욱 잦아지고, 또 32개의 센서를 시분할로 처리할 예정이어서 거기에 필요한 시간도 상당했다. 더우기 루프 테스트를 하면서 속도를 줄이지 않으려면 그 계산이 매우 빠르게 이루어져야 하고 최단시간 경로의 계산도 어려울 것으로 예상되었기 때문에 Speedy-3을 처음 개발할 때부터 Z-80A는 고려대상에서 제외되었다. 또 마이크로 로보트의 특성상, 기판 면적이나 전류소모등에 심한 제약이 있으므로 주변회로가 복잡한 일반 i8086, MC68000 등의 16비트 CPU는 역시 제외되었다. 따라서 구입 가능성까지 생각해서 사용을 고려한 CPU는 HD64180, V40, 8096 등이었다.

HD64180은 히다찌에서 만든 Z80 호환 CPU로서 Z80B보다 약간 더 빠르고, CMOS로서 전력소모가 적으며 여러가지 주변회로를 내장한 CPU인데 Z80과 명령어가 같으므로 사용 경험상 유리하지만, 그 처리 능력이 아무래도 모자랄 것 같아서 제외되었다. 8096은 인텔사의 16비트 프로세서인데 원래 고성능의 제어용으로 만들어진 것이어서 명령어의 구조와 내장하고 있는 I/O 디바이스 등이 마이크로 로보트에서 사용하기에 매우 좋게 되어 있으나 아무래도 익숙하지 않는 CPU라서 불안했고, 개발장비 등의 이용도 어려웠기 때문에 포기했다. 국민 보급형 PC에 사용되어서 유명해진 NEC의 V40은 8086과 명령어 수준에서 호환성이 있기 때문에 IBM-PC의 사용 경험을 이용하면 프로그램이 어렵지 않을 것으로 판단되었고, CMOS여서 전류소모가 적은데다가 그 처리속도는 8MHz 80188과 같은 수준이어서 충분할 것으로 보였다. 물론 제어용 프로세서가 아니므로 꼭 적합하다고 볼 수는 없으나 여러가지 사용상의 잇점을 고려하여 V40을 사용하기로 결정했다.

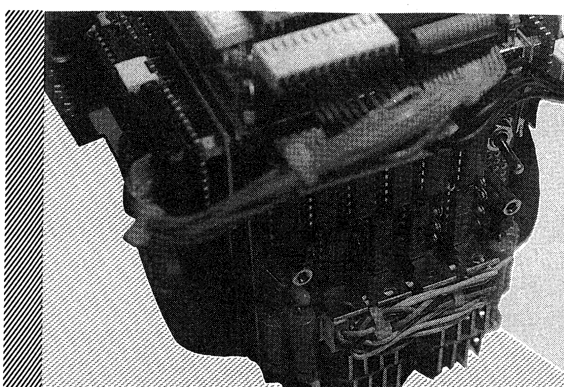
Turbo-4의 경우에는 Z80A와 함께 8031이 같이 사용되었는데, 어쨌든 고성능의 마이크로 로보트에는 Z80A가 조금 부족한 것이 아닌가 생각된다. 값은 좀 비싸지만 V40 등을 사용하면 IBM-PC의 성능 좋은 하이 레벨(High Level) 언어도 사용할 수 있다는점과 회로가 오히려 더 간단해지고, 전류 소모도 Z80A보다 더 적아진다는 점을 고려하면 굳이 성능 때문만이 아니라도 개발의 용이성 등을 고려해도 Z80A가 더 유리하다고 할 수 없는 것이다.

일단 사용할 CPU의 종류가 결정되고 나면 그 다음은 메모리와 I/O가 남는다. 경험상 ROM은 8KB, RAM은 2KB면 된다는 것을 알고 있었는데, 16비트 코드의 경우 그 길이가 얼마 늘어날지 몰랐고 또 하이레벨의 언어를 사용할 경우는 특히 오브젝트 코드(Object Code)의 크기가 커지므로 여유있게 ROM을 32KB까지 가능하도록 했고 RAM은 테스트 단계에서 프로그램도 RAM에 넣어서 사용할 것이었기에 역시 여유있게 32KB까지 쓸 수 있도록 했다. 실제로 사용한 IC는 27C64/27C256과 6264/62256이었는데 모두 핀 접속이 비슷하므로 간단히 점퍼(Jumper)에 의해 선택 사용이 가능하도록 했다. RAM은 액세스 타임이 120ns로서 8MHz V40에서 사용하기에 충분히 빠르지만 ROM은 200ns로서 거의 여유가 없었다. 따라서 ROM을 액세스할 때는 각 하나씩의 웨이트(Wait)가 삽입되었는데, 이는 V40 내부의 웨이트 스테이트 제너레이터에 의해 발생되었으므로 이에 따른 추가의 하드웨어는 없다.

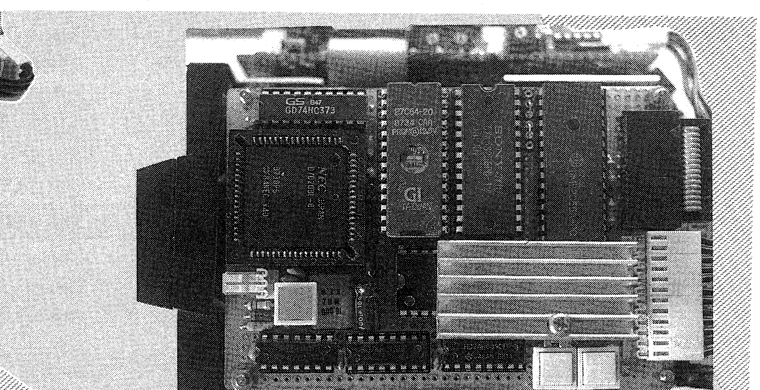
참고로, V40은 리세트된 후에는 무조건 웨이트를 발생하도록 되어 있으므로 프로그램의 시작부분에서 웨이트 제너레이터를 다시 프로그램하면 된다.

I/O는 82C55 하나면 여유가 있으

〔사진 5〕 몸체 뒤쪽의 모터 드라이버 보드



〔사진 6〕 CPU 보드



므로 별 문제가 없었는데, 카운터의 경우 양쪽 스텝모터에 각각 하나씩, 초음파 센서의 시간 측정용으로 하나, 주기억으로 인터럽트를 걸기위해 하나 등, 최소한 4개가 필요하여 V40 내부에 있는 것만으로는 부족하므로 8253을 외부에 하나 더 사용했고, 도합 6개이므로 나머지 둘 중 하나는 RS-232C의 보오 레이트 제너레이션(Baud Rate Generation)에, 또 하나는 압전 스피커를 구동하는데 썼다. 초음파 센서용으 40kHz 신호는 CPU의 8MHz 클럭을 분주해서 얻었고 이 중에서 2MHz의 신호를 얻어 스텝모터용으로 사용했다. CPU에 Z80을 사용하는 경우 스텝모터의 펄스 발생용으로 Z80 CTC를 사용하는 것을 볼 수 있는데, 고속 마이크로 로보트의 경우에는 적합하지 않다. 그 이유는 다음과 같다.

마이크로 로보트가 미끄러지지 않고 출발하기 위해서는 초기 속도가 대략 10cm / sec 이하여야 한다. 예를 들어 5cm / sec로 출발한다고 하자. 한 스텝당 1mm를 움직인다고 하면 50Hz가 된다. Z80 CTC는 8비트 카운터를 가지므로 입력 주파수는 50Hz×256=12.8kHz이하가 되어야 한다. 그 이상이면 100Hz를 출력할 수가 없다. 12kHz라고 하고 1cm / sec, 즉 100Hz를 발생시키려면 시상수(Time Constant)가 12kHz / 1000Hz=12가 된다. 여기서 가속을 하려면 그 다음 속도는 12kHz / 11 = 1090. 91Hz가 된다.

즉 한번에 변화하는 속도의 양이 커서 부드러운 가속을 할 수가 없게 된다. 스텝모터는 특히 고속에서 토크가 적으므로 갑작스런 속도의 변화는 탈조의 원인이 된다. 결과적으로 속도를 낼 수가 없다. 이를 피하기 위해서는 8253 등과 같은 16 비트의 카운터를 사용해야 한다.

사진 6에서 보는 바와 같이, CPU 보드의 나머지 회로는 특별한 것은 없다. 혹시 Speedy-3의 실물을 직접 보았거나 사진을 본 사람 중에 어드레스 래치가 74HC373 하나뿐인 것을 이상하게 여긴 이가 있을지 모른다. 실제로는 두개가 있는데, 하나는 RAM의 소켓 속에 있다. 그것은 처음 설계할 때, 착각을 해서 하나면 된다고 생각했는데 나중에 추가하려고 보니까 기판에 여유가 전혀 없어서 그렇게 된 것이다. 각 IC는 가능하면 CMOS를 사용하여 소비전력을 줄이도록 했는데 완성된 CPU 보드의 동작 전류는 200mA 정도였다. 이 보드의 전원은 모터용으로 전원을 7805를 사용하여 전압을 낮추면서 모터에 의한 전압 변동을 제거하여 이용했다. 메모리와 I / O 포트의 할당은 표 2와 같다. 참고로 CPU 보드의 전체회로도는 그림 3과 같다.

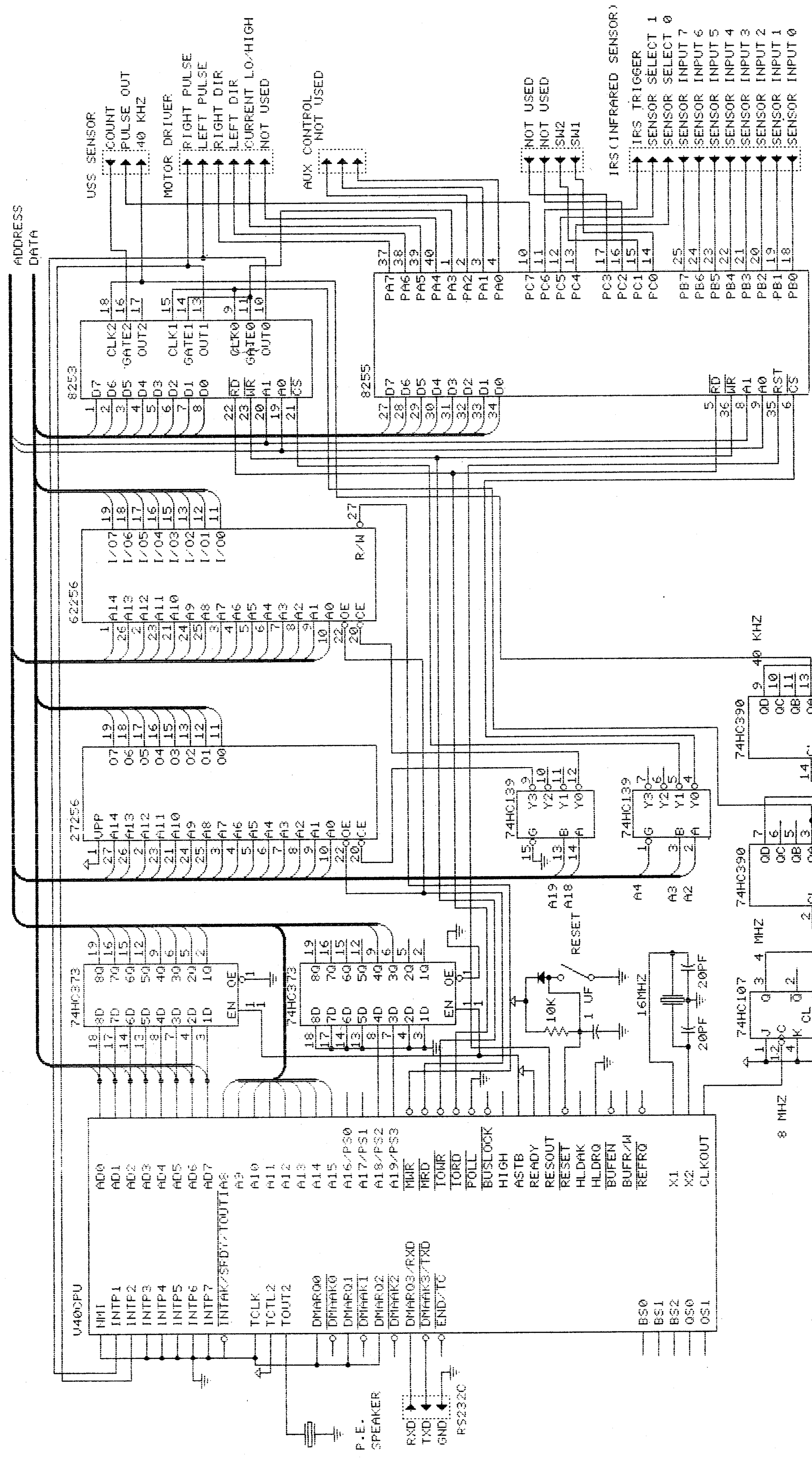
적외선 센서

마이크로 로보트가 안정된 주행을 하기 위해서는 옆 벽의 유무뿐만 아니라 옆벽까지의 거리도 꽤 정확히 알아서 그때 그때 자세보정을 해야 한다. 고속 주행을 위해서는 접촉식의 센서가 부적합한 것은 당연하므로 비접촉식 센서 중에서 선택해야 한다. 흔히 쓰이는 것은 적외선 센서와 초음파 센서이다.

〔표 2〕 메모리와 I/O 포트의 할당

메모리
0000 : 0000~0000 : 7FFF : RAM
F000 : 8000~F000 : FFFF : ROM
I/O 포트
0000~0003 : 8255
0004~0007 : 8253
0010~0013 : Interrupt Controller ;
0014~0017 : Timer Unit ; V40 내장
0018~001B : Serial Communication Unit ;
8255의 포트 할당
Port A (Output) – Connected to motor drive unit
PA0~PA2 : Not used
PA3 : 8253 gate control
PA4 : Not used
PA5 : 전류 제한 저항 on/off
PA6 : Left motor direction control
PA7 : Right motor direction control
Port B (Input) – Connected to IR sensor unit
PB0~PB7 : Infrared sensor input 0~7
Port C (Input/Output)
PC0~PC1 : Start/mode switch
PC2~PC3 : Not used
PC4~PC5 : IR sensor select
PC6 : IR sensor trigger
PC7 : Unltrsonic sensor trigger
8253 채널 할당
Channel 0 : Left motor – interrupt 1
Channel 1 : Right motor – interrupt 2
Channel 2 : Ultrasonic sensor timer
TULA (V40 on-chip timer) 채널 할당 (8253과 구분하기 위해 Ch.3~Ch.5로 번호를 매김)
Channel 3 : Timer interrupt – interrupt 0 ; 4kHz
Channel 4 : Connected to SULA (Baud rate generation)
Channel 5 : Connected to piezo-electirc speaker

〔그림 3〕 CPU 보드의 회로 구성



적외선 센서는 다시 반사형과 투과형으로 나뉜다. 이 중에서 옆벽 센서로서 초음파 센서는 먼저 고려 대상에서 제외되었다. 그 이유는 다음의 세가지이다.

첫번째로 초음파의 확산때문에 벽이 끝나는 점을 정확히 알 수 없다. 벽이 끝나는 점을 정확히 알아야 내부의 거리 변수를 그에 맞춰 보정해서 그 다음 구간의 정확한 중앙에 정지할 수 있고 그래야만 회전후 진로가 정확하게 된다.

둘째, 옆 벽까지 거리를 정확히 알 수가 없다. 일반적으로 마이크로 로보트에 사용되는 초음파 센서는 40kHz의 공진주파수를 갖는다. 음속이 340 m / sec÷40kHz÷2=4.25㎜이다. 이는 이상적인 값이고, 실제로는 이보다 더 큰 값이 될 수 있으므로 적합하지 않는다.

위의 두가지 문제는 높은 주파수의 초음파 센서를 이용하면, 어느 정도 해결된다. 그러나 옆 벽으로부터의 반사파를 받기 위해서는 초음파 송수신기를 벽 높이에 설치해야 하므로 설치할 수 있는 장소에 많은 제약을 받고 특히, 몸체에서 앞쪽으로 많이 떼어놓을 수가 없게 된다.

몸체에서 앞쪽으로 많이 떼어 놓을 경우 앞쪽 구간의 정보를 받아 들이는 시점이 늦어지고, 그때까지는 계속 직진할지 일단 정지해야 할지를 알 수 없으므로 속도를 낮춰두어야만 한다. 다시 말하면 앞 구간의 벽 정보를 얼마나 빨리 얻는지에 의해 1차 주행의 최고 속도가 결정된다.

투과형 적외선센서는 반사식보다는 더 확실한 동작을 하지만 역시 설치장소의 제약 때문에 적합하지 않았다. 이번 대회부터는 1차 주행의 시간이 기록에 영향을 주기 때문에 (개정된 규정을 참조) 1차 주행의 속도를 최대한 빠르게 해야 했고, 따라서 반사형 적외선 센서를 사용하기로 했다. 반사형 적외선 센서의 위치는 미로의 한 가운데에 정지했을 때 앞쪽 벽과 폴(Pole)을 감지할 수 있는, 즉 몸체의 중앙에서 정확히 9㎝ 떨어진 곳에 배치하기로 하고 또 곡선회전에 대비해서 몸체의 중앙부분에도 설치했다.

작은 오차라도 빨리 검출해야 보정이 쉽기 때문에 센서를 조밀하게 배치했는데 이 센서가 커버하는 영역이 좁으면 위험하므로 8개의 센서를 한 조로 하기로 했다. Turbo-4의 경우에도 마찬가지의 센서 배치를 하고 있으나 한 조에 3개씩으로 일단 큰 오차가 발생하면 센서의 감지 범위를 벗어나서 어쩔 수가 없다는 문제가 있었다. 어쨌든 도합 32개의 센서를 사용하기로 했으므로 그 회로를 어떻게 하면 간단하게 하는가 하는 것이 가장 큰 문제였다. 가장 간단히 하려면 센서의 출력을 바로 논리회로에 연결해 버리면 되지만 주변의 밝기에 많은 영향을 받으므로 적절하지 않다(참고로 대회에 이미 참가해본 사람은 알겠으나 대회장은 비디오 녹화 등의 이유로 매우 밝게 조명되어 참가자의 요청에 의해 이를 조정하지 않는다.) 결국 Speedy-3에 사용한 회로는 Speedy-2에 사용했던 것과 거의 동일한 회로인데 4조의 센서를 시분할로 구동하기로 했다. 그림 4를 참조하기 바란다.

각 조에서 나온 신호는 DC 성분이 제거된 후, 다른 조의 해당 센서와 함께 다이오드에 의해 Wired-OR가 된다. 즉, 가장 높은 전압이 출력된다. 이 전압은 비교기에 의해 기준 전압과 비교되고 그 결과가 74HC374에 저장된다. CPU는 4kHz, 즉 250㎲마다 인터럽트가 걸려 이전에 선택되었던 센서의 상태가 374에 저장되어 있는 것을 읽어 메모리에 저장하고 그 다음 센서를 선택하여 그 쪽의 적외선 LED를 약 100㎲ 동안 켜주게 된다. 이 시간은 74HC123에 의해 발생되고 CPU는 74HC123을 트리거한 후 바로 인터럽트 처리를 마친다. 이러한 회로에 의해 최소한의 부품을 사용하여 32개의 센서 모두를 조명에 거의 영향을 안 받도록 했다.

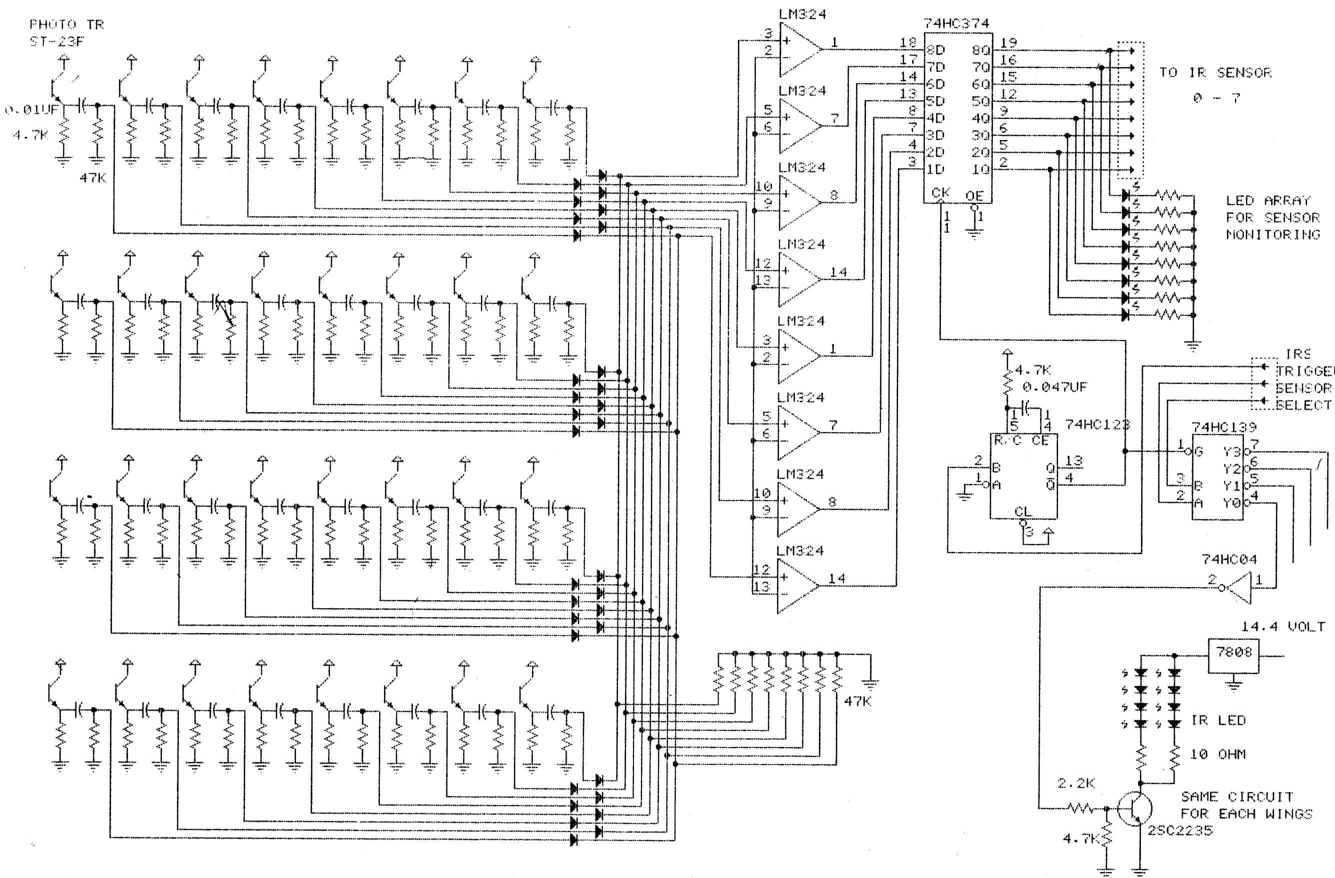
사진에서 보는 바와 같이 센서는 그 회로 뿐만이 아니라 설치 방법도 매우 중요한데 플라스틱 커버에 의해 외부의 빛을 막았고 밀접히 배치되어 있는 센서의 상호 영향도 칸막이에 의해 배제했다. 사실, 센서의 동작은 회로보다도 이러한 요소들에 더욱 좌우된다. 실험 과정에서는 센서 상태를 직접 눈으로 보고 알 수 있어야 하는데 8개의 LED를 사용하여 한번에 한 조의 센서의 상태만을 표시하도록 했다. 따라서 정상적인 주행중에는 여러 센서의 상태가 번갈아 표시되므로 알아볼 수가 없는데 실용상 지장은 없다.

여기서 사용된 포토 트랜지스터와 적외선 LED는 각각 ST-23F(H)와 EL-23F(H)라는 것인데 별로 특별한 것은 아니다. 보통의 규격을 가진 것을 사용하면 되는데 단지 주의할 것은 빔 앵글(Beam Angle)로서 ±10-13° 정도의 것을 사용하면 된다. 너무 각도가 좁으면 조정하기가 까다롭고 흔들림의 영향을 많이 받으며, 너무 넓으면 외란광의 영향을 많이 받게 되고 감도가 떨어진다. 또 빔 앵글과는 관계없이 에폭시 몰드(Epoxy Mold) 타입의 센서는 어느 방향으로 부터든지 강한 빛에는 영향을 받으므로 반드시 커버 해서 윗쪽으로부터의 조명을 막아야 한다.

초음파 센서

먼저 얘기했듯이 1차 주행의 최고 속도는 얼마나 빨리 앞쪽 구간의 벽에 대한 정보를 얻는지에 달렸다. 따라서 전방의 벽의 유무를 얼마나 빨리 알 수 있는가 하는 것이 매우 중요하다. 대회 규정에 의해 마이크로 로보트의 크기는 가로 세로 25㎝ 이하로 제한되어 있으므로 전방의 벽을 빨리 감지하기 위해서는 반사형의 센서를 사용해야 한다. 적외선이나 기타 빛에 의한 센서는 비교적 간단하지만 오동작의

〔그림 4〕 적외선 센서부분의 회로 구성



가능성이 크다. 먼 거리에 대해 동작해야 하는데다가 그 위에 외란광 차단시설을 할 수도 없기 때문이다. 따라서 사진 8에서 보는 것과 같은 초음파 센서를 사용하게 된다. 여기서 초음파 센서가 해야 할 일은 무척 간단하다. 전방의 정해진 위치에 벽이 있는지의 유무만 알면 되는 것이다. 굳이 전방의 벽을 이용해서 거리 보정을 해야할 이유가 없으므로 그 거리도 정확히 알 필요가 없다.

Speedy-3에서는 Speedy-2와 같은 초음파 센서 회로를 사용했는데, 여기에 특이한 것은 없다. 120ms마다 CPU에서 신호를 주면 74HC123에 의해 1ms 폭의 신호가 만들어짐과 동시에 RS FF이 세트되어 8253의 채널 2 카운터가 시간측정을 시작한다. 수신회로에서 반사파를 검출하면 RS FF을 리스트하여 시간측정을 끝내고 나중에 CPU에서 그 카운터의 값을 읽어들인다. 그 값이 일정범위 사이이면 특정 위치에 벽이 있다고 보는 것이다. 초음파 센서를 이 정도의 용도로 사용하는 것은 생각보다 그다지 어렵지 않지만 몇가지 주의가 필요하다.

첫째, 송신부에서 발생된 초음파가 직접 수신부로 들어가는 것을 막아야 한다. 여기에서는 작은 커버를 씌움으로써 해결했다.

둘째, 측정 주기를 너무 빨리 하면 그 전 주기에 발생된 초음파가 매우 멀리 있는 벽에 반사되어 돌아오는 것을 이번 주기의 것으로 오인할 수가 있다.

즉, 존재하지 않는 벽의 허상을 보게 되는 것이다. 실제로 실험중 측정 주기를 200Hz로 했을 때 그런 문제가 발생했었는데 처음에는 알고리즘상의 문제인 것으로 착각했었다. 이런 몇몇 문제는 그 해결이 쉽지만 발견은 어렵다. 충분히 실험을 하는 수 밖에 없을 것이다.

그림 5에서 보는 것과 같이 초음파 센서 회로에는 OP 앰프 등 아날로그 회로가 사용되므로 5V보다는 좀 높은 전압이 바람직하다. 특히 송신파의 강도를 크게 하기 위해서는 초음파 트랜스듀서의 양단에 높은 전압을 인가할 필요가 있고 그러기 위해서는 높은 전원 전압을 사용하거나 승압 트랜스를 사용해야 한다. 여기에선 모터에서의 노이즈를 차단하기 위해 일단 5V로 낮춰진 전원을 다시 15V로 승압해서

썼다. 전력 효율의 관점에서 볼때 상당히 비효율적이지만 어차피 초음파 센서가 소비하는 전력은 다른 것에 비해 작기 때문에 크게 문제되지 않는다.

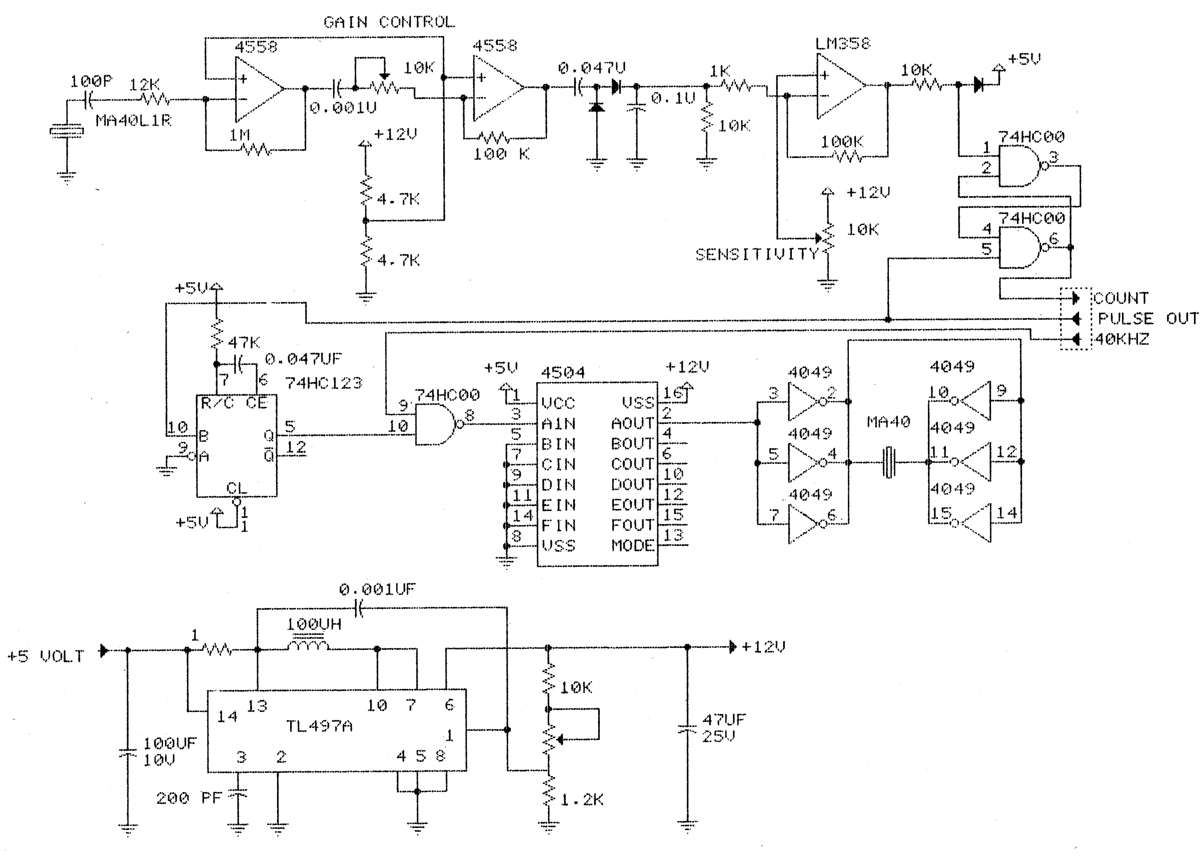
전원과 기타 회로

처음에는 전원으로서 7.2V 1300mA의 Ni-Cd 축전기를 사용하고 스텝모터의 구동은 정전류형 승압회로를 사용하려 했었다. 가까스로 좁은 면적속에 필요한 용량의 승압회로를 만들어 넣을 수 있었으나 잡음방지를 위한 회로를 넣을 공간이 없었기에 바로 가까이 위치한 논리회로가 오동작하는 것을 막을 수가 없었다. 결국 AA 사이즈의 Ni-Cd 축전기를 12개 직렬로 접속하여 14.4V 500mA로 만들어 사용했다. 이 전원으로 Speedy-3은 10-20분 정도 동작할 수 있었는데 대회 규정시간이 15분이고, 예상 기록등을 고려하면 충분하지만 계속되는 실험을 할 때에는 문제가 된다.

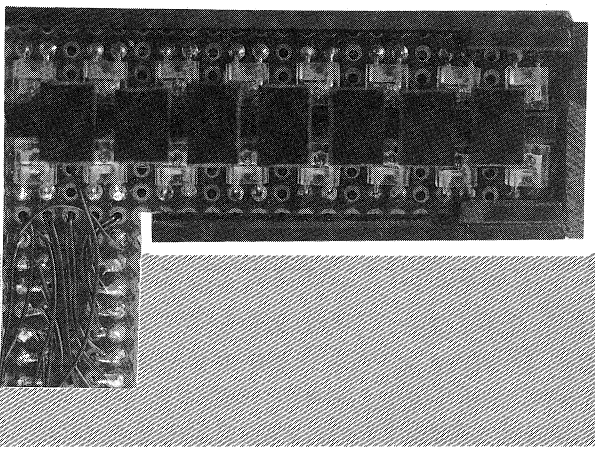
저속의 마이크로 로보트는 전원을 외부에서 공급해서 실험할 수 있으나, 고속의 경우에는 전선이 매달리는 것도 많은 영향을 주므로 축전지를 여러조 준비하고 실험하는 수 밖에 없다. 가장 좋은 것은 호스트 컴퓨터에서 가능한 많은 시뮬레이션을 행하는 것일 것이다. 필자는 미로탐색 알고리즘에 대해 시뮬레이션을 했는데 후에 설명할 루프 테스트 등의 구현에 있어서 많은 도움을 받았다. 곡선회전이나 오차의 보정 알고리즘 등도 가능하면 시뮬레이션 등을 통해 미리 실험해보는 것이 좋을 것이다. 특히 개인적으로 마이크로 로보트를 통해 미리 실험해 보는 것이 좋을 것이다. 특히 개인적으로 마이크로 로보트를 만드는 경우라면 큰 미로를 준비할 수가 없으므로 복잡한 미로탐색 알고리즘이나 고속에서의 특성 등에 대해 실제로 실험하기가 어렵다.

시스템의 규모가 커질수록 체계적인 디버깅 방법이 요구된다. 마이크로 로보트는 그 규모가 크다고 할 수는 없으나 여러 다양한 부분이 밀접하게 맞물려 있고 디버깅을 위한 여분의 하드웨어를 설치하기가 곤란하기 때문에 디버깅이 매우 어렵다. Speedy-2에서는 전용의 인터페이스 카드를 제작하여 IBM-PC에서 로보트의 시스템 버스를 직접 제어하는 방법을 사용했었다. Speedy-3에서는 사용한 V40 CPU에 직렬통신 회로가 내장되어 있기 때문에 그것을 사용했고 PC의 RS-232C

〔그림 5〕 초음파 센서부분의 회로 구성



〔사진 7〕 적외선 센서



〔사진 8〕 초음파 센서



와 연결하기 위해 MC1488 / 89에 의한 레벨 콘버터를 사용했다. 물론 이 회로는 Speedy-3와는 별개로 외부에 설치되었다.

기계적인 부분의 제작

사실, 전자공학 등을 전공하는 사람이 마이크로 로보트를 제작할 때에 가장 곤란을 겪는 부분이 바퀴나 동체 등의 부분이다. 또, 이 부분은 흔히 중요하지 않은 것으로 무시되기도 하는데 실제로는 매우 중요하다. Speedy-3의 동체는 주로 1㎜ 두께의 알루미늄 판을 가공해서 만들었다. 외란광의 차단을 위한 센서 덮개 등에 2㎜ 두께의 아크릴 판을 썼고 부분적으로 에폭시 기판 자체를 구조재로서 사용한 곳이 있다.

로보트의 아래부분의 폭은 가능한한 좁은 것이 좋으므로 두 모터는 완전히 밀착시켜 배치했고 바퀴도 바싹 붙였으며 여분의 축은 잘라버렸다. 바퀴는 알루미늄재로서 주문에 의해 선반 가공을 해서 만들어진 휠에 고무판을 잘라 감아 붙였다. 마찰력은 그럭저럭 괜찮은 편이다. 고무 부분의 단면은 직사각형으로 지금 생각해 보면 바퀴가 땅에 닿은 부분의 폭을 좁게 해야 회전 오차가 적을 것 같다. 또 대회를 준비하는 측에서는 가능한한 미로의 바닥을 깨끗이 유지하려 하지만 아무래도 한참 주행을 하고나면 바퀴에 먼지가 끼어 마찰력을 현저히 감소시킨다. 가능하면 진짜 자동차의 타이어처 럼 홈을 만드는 것도 좋을 것이다. Speedy-3에서는 먼지터는 솔을 바퀴에 닿도록 해 놓을 것을 진지하게 고려했으나 고속 주행시 방해가 될것 같아서 사용하지 않았다.

가장 까다로운 부분은 앞 / 뒤 보조바퀴이다. 회전 관성을 줄이려면 무게 중심을 가능한한 중심에 가까운 곳에 두어야 한다. 따라서 가 / 감속시 각각 뒤 / 앞으로 쓰러지는 것을 막기 위해서는 양쪽에 보조바퀴를 달아야 한다. 가운데 바퀴가 항상 땅에 닿아야 하므로 두 보조 바퀴중 하나는 반드시 땅에서 떨어져 있어야 한다.

예를 들어 앞 보조 바퀴가 땅에 닿아 있다고 하자. 출발시의 급격한 가속때문에 로보트는 뒤로 기울어지게 되고 떠있던 뒤 보조바퀴가 땅에 닿게 된다. 이때 닿은 순간의 충격을 부드럽게 흡수하지 못하면 자칫 방향과 거리 오차의 원인이 된다. 완충장치를 달면 좋겠으나 위아래로 흔들리는 정도가 심하면 측면 적외선 센서의 동작에 영향을 줄 수도 있다. 특히 보조 바퀴는 여러 방향으로 구를 수 있어야 하므로 일반적인 캐스터(의자 밑에 달린 바퀴 같은 것)를 사용하면 방향이 전환되는 순간에 상당한 흔들림이 유발된다. 볼 캐스터를 사용하면 되지만 소형이고 양질의 것은 국내에서 구할 수가 없다.

Speedy-3에서도 이 문제를 완전히 해결할 수는 없었는데 특수한 자작 캐스터를 사용했다(사진 8 참조). 이에 의해 대부분의 경우에는 문제가 없었는데 후진에는 심각한 영향을 주어 결국 후진은 포기할 수 밖에 없었다(막힌 곳을 돌아 나올때 그 다음 회전할 곳까지 그대로 후진을 하면 시간을 절약할 수 있다.) Speedy-3의 뒷 보조바퀴는 가속시에만 뒤로 기울어져 땅에 닿으므로 앞쪽으로만 구를 수 있는 구조로 되어 있다. 보조 바퀴를 선택 / 제작할 때에는 이같은 점들 외에도 지나치게 직경이 작은 바퀴를 사용하는 것을 피해야 하고, 가능하면 완전히 딱딱한 바퀴는 피해야 한다. 그 이유는 대회용의 미로는 4장의 미로가 연결되어 만들어지고 그 연결부위에 약간의 요철이 있기 때문이다.

마지막으로 마이크로 로보트의 동체, 특히 전면부는 튼튼하게 만들어야 한다. 실험도중 무수히 많이 충돌할 것이

기 때문이다. 만일 공간적인 여유가 있으면 충격 흡수식의 범퍼라도 달면 좋을 것이다. 개발 과정에서는 여러 보드를 손볼 때가 많으므로 각 보드가 노출되도록 하든지 또는 쉽게 개폐가 가능하도록 하는 것이 좋다. 부서지기 쉬운 부분이 있다면 여분을 준비해야 대회때 당황하는 일이 없을 것이다. 또 최고 속도 1-2m / sec 정도에서 굳이 유선형의 차체가 필요한 것은 아니지만 열이 많이 나는 부품은 바람에 노출되도록 배치되는 것이 좋다.

하드웨어의 개선점

여러대의 마이크로 로보트를 개발해 오면서 느낀 것은 마이크로 로보트의 성능은 한대 한대에 들인 시간과 노력보다도 몇대째 만드느냐 하는 것에 많이 좌우된다는 것이다. 이 점에 대해서는 Turbo-4의 제작자인 박남규씨도 동의한 바 있는데, 마이크로 로보트가 일단 제작에 들어가면 처음에 구상한 것을 바꾸기가 어렵기 때문이다. 물론, 작은 변경이야 언제든지 가능하겠지만, 대회에 임박해서 전체적인 구조가 잘못되었다든지, 센서의 선택을 잘못했다든지, 센서의 선택을 잘못했다든지 하는 것을 깨달았더라도 그것을 변경할 시간적인 여유가 없는 것이다. 더우기, 사람들이 하는 일이 흔히 그렇듯이 항상 대회 몇주 전에야 열심히 하게 되므로 매번 로보트를 만들고, 대회를 치룰 때마다 다음에 다시 만들 때는 이런 저런 점을 개선해야지 하는 아쉬움을 갖게 된다.

그런 점에서 볼 때, Speedy-3의 경우에는 하드웨어에 대해서는 불만이 적었다. 모터는 기대했던 정도의 성능을 발휘했고, 다른 회로들도 모두 안정되게 기능을 수행했다. 부족했던 것은 소프트웨어로서, 열심히 만든 하드웨어의 기능을 충분히 발휘하지 못했다는 생각이 든다. 이에 대한 구체적인 것은 소프트웨어의 설명을 할때 함께 말하겠다. 그러나, 완벽한 것이란 없고 또 Speedy-3에서 취한 방식이 가장 좋은 방식이라고 생각하지는 않기 때문에 Speedy-3을 제작하면서, 대회를 치루면서, 또는 그 후에야 생각이 났던 점들에 대해 여기서 언급해 두는 것이 앞으로 이 대회에서 좋은 기록을 내고자 하는 이들에게 도움이 될 것이다.

첫째, 가속성능은 파워 / 무게 비율과 바퀴, 무게중심에 의해 결정된다. Speedy-3과 Turbo-4의 파워 / 무게 비율은 대체로 비슷했다. Speedy-3의 전체 무게가 2kg 정도로서 Turbo-4의 두배 정도 되었고, 토크는 두배를 약간 넘었다. 따라서 직진 가속 성능은 Speedy-3가 Turbo-4보다 약간 우수했으나 큰 차이는 없었다. 여기서 유리해지기 위해서는 물론, DC 서보모터를 사용하면 된다. 그러나 스텝모터를 사용하면서 파워 / 무게 비율을 개선하려면 전자회로쪽의 무게를 줄이는 수밖에 없다. 회로가 간단해지면 무게가 줄어듬과 동시에 무게중심도 낮아지므로 더욱 유리하다.

필자는 대회가 끝난 후, 8031에 대해 검토한 결과 8031 하나로도 마이크로 로보트의 제어가 가능하다는 결론을 얻었다. 단, 내장된 카운터의 수가 적으므로 80C252 등을 사용해야 할 것이다. 어차피 대용량의 RAM이 필요하게 되는데, 프로그램을 ROM에 넣지 않고 RAM에 함께 넣은 후 외부에서 DMA에 의해 프로그램과 데이터 모두를 넣을 수 있다. 이렇게 하면 CPU 보드가 무척 간단해지고 전력 소모도 줄어들게 된다.

둘째, 고속의 마이크로 로보트에서 보조바퀴의 역할은 아무리 강조해도 지나치지 않다. 아주 좋은 품질의 볼캐스터를 구하지 못하면, 차라리 Speedy-3과 같이 돌아가게 되어있는 구조의 것을 사용하는 것이 낫다. 또 무게중심을 주 바퀴에 두는 이상 앞뒤 모두에 보조 바퀴를 달아야 하는데 가 / 감속시의 흔들거림은 주행오차의 가장 큰 원인의 하나이다. 필자는 개발 도중 무선조종 모형에서 사용되는 초소형의 서보모터를 사용해서 보조바퀴의 방향을 돌려주는 것을 고려한 일이 있다. 이 서보모터는 무게가 수십g 정도로서 크게 부담이 되지 않고 또 전력소모도 그다지 크지 않으므로 마이크로 로보트에도 여러가지로 이용할 수 있을 것이다. 뒤 보조 바퀴를 올렸다 내렸다 하는 것도 고려했다. 요즘은 무선조종 모형에 대한 열기가 많이 식어서 잘 모르는 사람이 많겠지만 여기서 사용되는 서보모터는 단일 전원(5-6V)과 1-2ms 폭의 신호를 주는 것만으로 위치 제어가 된다. 또, 이미 언급한 적이 있으나, 바퀴의 땅에 닿는 부분의 폭을 좁게하는 것이 좋을 것 같다. 넓다고 마찰력이 반드시 좋은 것도 아니고, 회전할 때 오차의 원인이 된다.

세째, 처음 시작할 때 부터 어떻게 디버깅을 할 것인지 대책을 세워두어야 한다. 파스칼이나 C 등의 고수준 언어를 사용하는 것도 좋을 것이다. 주행중인 로보트를 정지시켜 내부의 변수값을 확인할 수 있어야 디버깅이 간편해진다. 아울러, 가능하면 많은 부분을 시뮬레이션하는 것이 좋다. 아무래도 스크린 상에서 하는 것이 테스트용 미로에서 하는 것보다 쉽고, 또 좁은 미로에서만 테스트를 하면, 복잡한 미로에서 발생할 수 있는 논리상의 오류를 찾을 수가 없다. 특히 1차 주행의 알고리즘이 복잡해 질수록 많은 유형의 미로에서 시뮬레이션을 해야 한다.

네째, 너무 많은 보드에 회로를 분산시키지 않아야 한다. 보드 사이를 연결하는 코넥터만도 상당한 분량도 된다. 따라서 회로를 단순히 하기 위해 많은 노력을 기울여야 한다.

마지막으로, 주행 중 오차는 빨리 발견할수록 좋다. 여기서 빨리라는 말은 20ms보다 10ms전에 발견하는 것이 좋다는 뜻이 아니고 2㎜보다 1㎜의 오차가 발생했을 때 발견하는 것이 좋다는 뜻이다. 따라서 하드웨어가 지나치게

복잡해지지만 않는다면 센서의 수는 많을수록 좋다. 일단 오차가 커지면 그의 보정은 여러가지로 어려워지고, 충돌할 가능성이나 벽의 유무를 잘못 판단할 확률이 훨씬 커진다.

이상으로 간단하나마 Speedy-3의 하드웨어 제작에 대해 기술하였다. 부분적으로는 충분한 설명이 되지 못한 부분도 있으리라 생각한다. 지면 관계상 소프트웨어에 관한 설명과 프로그램 리스트는 다음 호에 게재하기로 하겠다. 아무쪼록 마이크로 로보트 제작에 관심있는 사람들에게 많은 도움이 되길 바란다. PC

〈참고문헌〉

1. 마이크로 로보트 입문, 도서 산업사.
2. MicroMouse, 도서출판 청암.
3. 스테핑 모타의 활용기술, 도서출판 세운.
4. Stepping motors and their microprocessor controls, Oxford University Press.
5. 전자과학 1988.6~8월 V40/50-NEC의 16비트 μProcessor.

하드디스크의 용량을 배가시키는

하드디스크는 점차로 컴퓨터 사용자들에게 기본적으로 구비해야할 장치로 인식되고 있다.

물론 이같은 하드디스크의 수요증가는 그동안 가격이 낮아진 데에 그 원인이 있지만 아직도 누구나 쉽게 구입할 정도의 가격수준이라고 보기에는 힘들다.

요즘의 PC업체에서는 대부분 판매가격에 해당하는 기본시스템에 20M바이트의 하드디스크를 포함시키고 있어서 일반 사용자들에게는 20M바이트 하드디스크가 널리 쓰이고 있다.

그러나 일반사용자가 계속적인 작업을 하다 보면 더 많은 용량을 필요로 할 것이고, 이를 커버하기 위해 적당한 용량의 하드디스크를 하나 더 설치하게 될 것이다. 이러한 경우 기술적인 문제는 둘째 치더라도 가격면에서 고민을 더하게 된다.

마침 이러한 시기에 기존 하드디스크의 용량을 최고 3배까지 증가시켜 주는 디스크더블러(DiskDoubler)가 국내에도 소개되었다.

디스크더블러는 Datran Corporation 사에서 1988년 8월에 개발되어 DD2000이라는 모델명으로 미국에서 판매되어 오다가 최근 국내에 들어온 제품이다. 적용 기종은 IBM-PC XT / AT, 386과 버전 2.0 이상의 도스를 채택하고 있는 호환기종들이다. 특히 국민보급형 PC나 하드디스크를 한대 이상 연결할 수 없는 컴퓨터에 적용한다면 그 효과가 더욱 크리라 본다.

구성

디스크더블러는 보드 하나와 관련 소프트웨어 화일들이 들어 있는 5.25인치 디스켓 1장으로 구성되어 있다. 디스켓 속에는 다음과 같은 화일들이 들어있다.

- **DDINSTAL.EXE**: 디스크더블러 소프트웨어들을 하드디스크에 인스톨 시킨다.
- **DD.COM**: 디스크더블러의 시스템 소프트웨어이며 RAM상주 프로그램(Terminate and Stay Resident 또는 TSR)이다.
- **DDIR.EXE**: 확장된 DOS 디렉토리 명령
- **DDCMP.COM**: 화일을 압축시킨다.
- **DDUNCMP.COM**: 원래 화일로 확장시킨다.
- **COPYC.COM**: 화일 압축과 함께 화일복사를 병행한다.
- **COPYU.COM**: 화일 확장과 함께 화일복사를 병행한다.
- **DDEXCLUD.COM**: 원하는 화일을 압축 대상에서 제외시킨다.
- **DDINCLUD. COM**: 원하는 화일을 압축대상에 포함시킨다.
- **DDCHECK.COM**: 디스크더블러의 상태를 체크하고 디스플레이 시켜 준다.
- **DDREMOVE.COM**: 메모리로부터 디스크더블러를 제거한다.
- **DDENABLE.COM**: 메모리에서 디스크더블러를 활성화시킨다.
- **DDISABLE. COM**: 메모리에서 디스크더블러를 억제시킨다.
- **DDCONFIG.EXE**: 드라이브의 압축/확장 설정에 대한 디폴트를 세트
- **DDSWITCH.EXE**: 보드에 알맞게 스위치 세트를 변경한다.

기능

디스크더블러는 다트란(Datran)의 고유 기술 즉, 데이터를 자동으로 압축해서 디스크로 기억시키고 또는 원래의 확장된 형태로 복원시켜 주기 위한 하드웨어/소프트웨어의 결합체이다.

그리고 HCMOS VLSI 칩이 디스크더블러 보드에 장착되어 있어서 이 두동작(디스크에 읽고, 쓰기)을 고속으로 실행시켜 준다. 그렇기 때문에 고속의 디스크를 쓰고 있는 사용자라도 디스크더블러 운영시 수행 손실이 거의 없게 된다.

일단 인스톨작업이 완료되면 디스크더블러와 관련 유틸리티들의 조작이 도스 및 그의 유틸리티들의 쓰임새와 거의 흡사함을 보게 된다.

디스크더블러는 워드프로세서, 스프레드 시트, 디베이스 등 비지니스 어플리케이션 프로그램의 데이터들을 최적화된 압축형태로 변형시키는 고유 기법을 이용하고 있으며, 압축형태도 화일의 타입에 따라 다양하다.

비단 데이터 화일뿐만 아니라 프로그램 화일이나 바이너리 화일 (.COM, .EXE등)도 압축 가능한 데 단지 이 경우는 낮은 압축비로 압축될 뿐이다 (표 1 참조).

실제로 PC에서 워드프로세서, 스프레드 시트, 디베이스 화일들을 예로 압축시켜 보면 압축비가 거의 2:1에 가깝게 된다. 나중에 설명할 DDIR 유틸리티를 이용하면 실행후에 실제 얻은 화일의 정확한 압축비를 알 수 있다.

〔표 1〕 시스템화일과 데이터화일의 압축비 비교

화일 종류	압축비 범위
워드프로세싱	1.5:1~2.5:1
스프레드 시트	1.3:1~2.5:1
dBASE 화일	1.5:1~8.0:1
COM과 EXE 화일	1.0:1~2.0:1

디스크더블러 보드 설치

디스크더블러는 시스템 내의 다른 보드나 디바이스들과 I/O 포트 어드레스가 서로 상충되지 않도록 하기 위해 4개의 딥 스위치(Dip Switch)를 마련해 놓고 있다.

표 2를 보면 I/O채널이 다른 요소들과 상충되지 않도록 가능한 한 많은 포트어드레스를 설정해 놓은 것을 볼 수 있는데 디스크더블러에 의해 모두 점유할 수 있다.

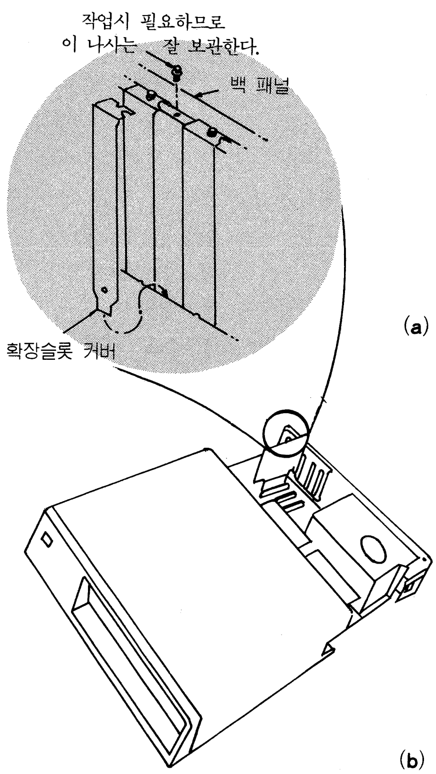
그러나 표 2에서 보듯이 디스크더블러 디폴트스위치 위치는 position 2에 세트되어 있음을 알 수

디스크더블러

〔표 2〕 설정가능한 보드와 스위치 세트

스위치 세팅 S1	S2	S3	S4	포트 어드레스	위치	설정 가능한 보드
on	on	on	on	200	1	비디오 게임 어댑터
on	on	on	off	220	2	디스크더블러
on	on	off	on	240	3	
on	on	off	off	260	4	LPT2
on	off	on	on	280	5	
on	off	on	off	2A0	6	
on	off	off	on	2C0	7	AST 클럭
on	off	off	off	2E0	8	COM 2
off	on	on	on	300	9	
off	on	on	off	320	10	
off	on	off	on	340	11	
off	on	off	off	360	12	LPT1
off	off	on	on	380	13	
off	off	on	off	3A0	14	모노크롬 어댑터
off	off	off	on	3C0	15	칼라 어댑터
off	off	off	off	3E0	16	디스크 콘트롤러/COM1

〔그림 1〕 디스크더블러 보드 설치



있다.

POSITION 2 :
SW1=ON,SW2=ON,SW3=ON,SW4=OFF

보드를 설치하기 전에 반드시 컴퓨터 전원을 끈다. 디스크더블러 보드의 딥 스위치를 위에서 보인 바와 같이 조정한다. 다음 컴퓨터 커버를 벗긴 후 쓰이고 있지 않은 확장 슬롯을 찾아 슬롯 커버를 제거한다. 이 때 조그만 나사(Mounting Screw)가 나오는데 이것은 디스크더블러 보드를 설치할 때 필요하므로 잘 보관해 둔다. 그 다음에 슬롯에 보드를 정확하게 장착한 후 PC 섀시를 고정시키고 컴퓨터 커버를 다시 고정한다(그림 1 참조).

이제 이 시스템은 실행될 충분한 준비가 된 셈이다. 전원을 연결하고 스위치를 ON시키면 컴퓨터는 평상시와 같이 부팅된다.

인스톨 프로그램

이제 디스크더블러 보드의 설치는 끝난 셈이고 남은 것은 DDINSTAL 프로그램을 이용해서 디스크더블러 소프트웨어를 설치하는 작업뿐이다.

여기서 DDINSTAL은 여러가지 일을 수행하고 있는데 이를 요약해 보면 다음과 같다.

1. 작업(인스톨) 대상 드라이브를 지정한다.(디폴트는 C)
2. 지정한 드라이브에서 디스크더블러를 위한 새로운 디렉토리를 생성한다.(디폴트 디렉토리는 "DD"이다)
3. FILES=25 및 디스크더블러의 디렉토리의 패스를 나타내는 AUTOEXEC.BAT화일을 만들어 시스템 부팅시 DD.COM이 자동으로 실행될 수 있도록 CONFIG.SYS를 수정한다.
4. 디스크더블러 디스켓속의 모든 화일을 지정한 드라이브의 디렉토리로 복사한다.
5. 이 프로그램은 또한 디스크더블러의 딥스위치 세트가 올바르게 되어 있는지를 체크해 준다. 만일 딥스위치 세팅이 시스템 구성과 어울리지 않게 되어 있으면 인스톨작업을 중단하고 사용자에게 스위치 세팅을 바르게 하기 위해서 DDSWITCH 유틸리티를 사용하라고 알려 준다.
6. 스위치 세트가 올바르게 되었다면 이제는 데이터를 자동으로 압축하여 저장하고 싶은 드라이브를 지정해야 한다.(디폴트 드라이브는 C이다.)
7 이제 작업할 드라이브만 지정하면 COM, EXE, BAT, SYS, BIN 화일들을 제외한 모든 화일들을 압축해서 저장한다.(처리시간은 XT의 경우 20M바이트 용량이 꽉 찼다고 가정했을 때 40분에서 60분 정도이다.)

인스톨 작업은 이와같이 간단하게 끝난다. 한가지 여기서 주의할 것은 인스톨명령을 내리기 전에 CONFIG.SYS의 내용을 확인하는 것이다.

즉 시스템 화일들은 별도로 지정하지 않는 한 원래 화일로 남아 있게 되지만 그 외의 화일 중에서 압축대상에서 제외하고 싶

은 것이 있으면 DDEXCLUD 유틸리티를 통해 대피를 시켜야 하는 것이다.

〈보기〉

1. 루트 디렉토리로 제어를 옮긴다.(여기서는 하드디스크 C를 작업대상으로 했다.)

```
C>CD\ ↵
```

이렇게 하면 어떠한 서브 디렉토리에서도 직접 루트 디렉토리로 제어를 옮길 수 있다.

2. CONFIG.SYS 화일의 내용을 보고 싶으면 DOS의 TYPE 명령어를 이용한다.

```
C>TYPE CONFIG.SYS ↵
```

화면에 디스플레이 된 내용들을 보면서 제외해야 할 화일이 없나 다시 한번 확인한다. 한 예를 화면 1에 나타내었다. 화면에서 보듯이 올바르게 시스템을 실행시키기 위해 CONFIG.SYS에서 필요로 하는 화일이 몇개 있다. 즉 DOS 서브 디렉토리에 있는 ANSI.SYS, DRIVER.SYS, DMDRVR.BIN과 TAPE 서브 디렉토리에 있는 TAPEDRVR.SLS가 그것들이다. 여기서 지정된 화일들은 시스템 화일들과 함께 Global Exclusion File속에 보관되어 자동 압축처리로부터 보호를 받게 된다.

계속해서 디스크더블러 시스템과 유틸리티 화일들이 들어 있는 디스켓을 드라이브 A에 넣고 "G_._"화일을 하드디스크의 루트디렉토리로 복사한다.

```
A>COPY G_._ C:\ ↵
```

인스톨되는 과정에서 TAPEDRVR. SLS의 압축을 피하기 위해 DDEXCLUD를 이용한다. 정확한 패스와 화일이름을 지정해야 한다.

```
A>DDEXCLUD
 C:\TAPE\TAPEDRVR.SLS ↵
```

그러면 DDEXCLUD는 TAPE 디렉토리 속에 EXCLUTION화일 "B_._"의 디렉토리를 만든다. A>DIR C: ↵를 입력해 확인해 보면 해당 화일이 TAPE 서브 디렉토리 속에 들어 있음을 알 수 있다.

마찬가지로 이번에는 아무런 화일이름의 지정없이 DDEXCLUD 를 실행시켜서 "B_._"화일의 상태를 살펴 본다.

```
A>DDEXCLUD C: ↵
```

그러면 \TAPE 서브 디렉토리 속에서 TAPEDRVR.SLS가 압축되지 않은 채 그대로 있음을 볼수 있다(화면 2 참조). 이제는 인스톨 준비가 다 된 셈이다. 즉 모든 CONFIG.SYS 화일의 수정작업이 끝난 다음 디스크더블러 시스템 화일과 유틸리티가 들어 있는 디스켓을 드라이브에 넣고 DOS 프롬프트상에서 DDINSTAL을 입력한다.

```
A>DDINSTAL ↵
```

처음에 인스톨메시지가 화면에 나타나고 작업을 계속하고 싶으면 아무키나 누르라는 메시지가 나온다(화면 3 참조).

키를 누르면 디스크더블러 소프트웨어를 설치하고자 하는 드라이브명을 묻는다.(디폴트는 C 드라이브)

```
Please select the drive you
want DiskDoubler Installed on.
(C,D,E,etc):_
```

다음에는 디폴트 디렉토리 이름으로 DD를 원하는지 묻는다.

```
Do you want to use "DD" as the
directory name for DiskDoubler?
Yes/No(Y/N) :_
```

〔화면 1〕 CONFIG.SYS 내용

```
C>TYPE CONFIG.SYS

 DEVICE=\DOS\ANSI.SYS
 DEVICE=\DOS\DRIVER.SYS /d:0 /f:0 /h:2
 DEVICE=\DOS\DRIVER.SYS /d:1 /f:2 /h:2
 DEVICE=\DOS\DRIVER.SYS /d:2 /f:5 /h:4 /n /t:612
 DEVICE=\DOS\DMDRVR.BIN
 DEVICE=\TAPE\TAPEDRVR.SLS
 BUFFERS=10
 FILES=25

C>_
```

〔화면 2〕 DDEXCLUD 화면

```
A>DDEXCLUD C:

 ????????.COM  has been Excluded from compression
 ????????.EXE  has been Excluded from compression
 ????????.BAT  has been Excluded from compression
 ????????.SYS  has been Excluded from compression
 ????????.BIN  has been Excluded from compression
 TAPEDRVR.SLS  has been Excluded from compression

A>_
```

이때 N(o)을 지정하면 원하는 디렉토리 이름을 다시 묻는다.

```
Please enter the directory
name of your choice:_
```

(만일 유저가 지정한 이름과 똑같은 디렉토리가 이미 있으면 계속 질문이 반복된다.) 이제부터는 디스크더블러의 인스톨이 자동적으로 수행한 후 작업을 마치게 된다.

그밖의 중요한 화일관리용 명령어들을 살펴보면 다음과 같다.

- **DDCMP**: 지정한 화일을 압축시킨다. 그러나 이 DDCMP 명령으로도 DDEXCLUD 화일을 이용해서 대피시킨 화일이나 이미 압축된 상태로 있는 화일은 더 이상 압축되지 않는다. 만일 이미 대피시킨 화일을 압축하려고 시도한다면 "excluded"라는 메시지가 출력된다.

대부분의 DOS명령과 마찬가지로 DDCMP도 와일드카드 문자를 이용하여 한번의 명령으로 여러 화일들을 연속적으로 압축시킬 수 있다. 물론 와일드카드 문자를 화일이름에 넣어 실행시킬 때 이미 압축된 화일을 만나면 그 화일은 무시하고 지나치게 된다. 입력 형식을 보면 다음과 같다.

```
C>DDCMP \Pathname\Filename.* ↵
```

〔화면 3〕 초기화면

```
DISKDOUBLER will automatically compress ALL files on your
hard disk(s) with the exception of all COM, EXE, SYS, BAT and
BIN files. They can be selectively compressed at a later time.
This feature greatly simplifies the installation and makes disk
space available for your immediate use.

It takes approximately 45 to 60 minutes to compress a full 20MB
drive on a standard IBM PC/XT running at 4.77 Mhz. The faster
your PC, the faster the compression.

Press any key when ready to continue:
```

〔화면 4〕 DDIR 실행화면

```
C>DDIR D:\COBOL
 DD RESIDENT IN MEMORY

  Disk Doubler DIRECTORY -- DDIR
  Directory of D:\COBOL

                Uncompressed   Compressed   Compression   Date       Time
Filename.EXT    File  Size     File  Size     Ratio       00-00-00   00:00a
------------    ------------   ----------   -----------   --------   ------
EDIT    .E!!          29696        26882           1.10   10-19-84    9:00a
H       .BAT              8            8           1.00    1-01-80   12:04a
HHH     .DAT         125312        88102           1.42    1-01-80   12:09a
HOI     .CBL          14726        10094           1.46    1-01-80   12:02a
PRI10   .COB           2304          921           2.50    1-01-80   12:44a
THP     .C!!          36142        25975           1.39    2-14-86    1:14p
------------    ------------   ----------   -----------
Files   9            208188       151982      Avg. 1.48

           Total Bytes available on disk: 18581504
```

DDCMP는 작업을 하는 동안 계속적으로 압축되고 있는 화일 이름을 화면에 나타내 준다.(마치 DOS에서 COPY 명령을 쓸 때와 비슷하다.) 또 COM, EXE, SYS, BAT 그리고 BIN 화일을 먼저 압축하고 싶으면 DDINCLUD를 이용하는데 이는 위에서 열거한 확장자들의 이름을 가진 화일들을 압축하고자하는 특정 디렉토리로 정의해야 가능하다.

이들 확장자명으로 된 화일을 압축시키면 압축된 사실과 DOS에 의해 실행시킬 수 없다는 것을 나타내기 위해 확장자명을 변경시켜 버린다. DOS는 이 압축된 화일을 실행시키지 못한다.

〈예〉

COMMAND.COM COMMAND.C!!
EXECUTE.EXE …… EXECUTE.E!!
BIN.BIN …………… BIN.B!N

그러나 DDCMP는 COM, EXE, SYS, BAT 그리고 BIN의 확장자명 자체는 변경시키지 않는다.

- **DDUNCMP**: 변형 축소된 화일을 원래 상태로 복원한다.

DDUNCMP는 DDCMP 프로그램과 상반되는 역할을 하는 프로그램으로서 조작방법은 DDCMP와 비슷하다. 이것도 코맨드 라인 프로그램으로 DDCMP에 의해 변형처리된 화일을 원래대로 복원시켜 준다. 그리고 DDUNCMP는 DDINCLUD 또는 DDEXCLUD된 것이라도 개의치 않고 화일을 원상 복귀시킨다. DDCMP처럼 DDUNCMP도 와일드 카드 문자를 사용할 수 있다.

만일 변형처리된 COM, EXE, SYS, BAT 또는 BIN 형태의 화일을 DOS에서 쓰고자 할 때에는 반드시 DDUNCMP 유틸리티를 이용해서 정상적인 화일 형태로 만들어 놓아야 한다. DDUNCMP는 자동적으로 DOS가 인식할 수 있도록 원래의 확장자명으로 돌려서 기억한다.

〈예〉

COMMAND.C!! COMMAND.COM
EXECUTE.E!! …… EXECUTE.EXE
BATCH.B!! ……… BATCH.BAT
SYSTEM.S!! …… SYSTEM.SYS

- **COPYC**: 병형처리 작업과 함께 복사작업이 진행된다.

예를 들어 다음과 같이 입력하면 D드라이브의 지정 패스내의 모든 화일을 압축해서 A드라이브로 기억시킨다.

```
C>COPYC D:\Pathname\Filename.*
  a: ↵
```

이와 비슷한 명령으로 COPYU가 있는데 역할만 다를 뿐이다.

그외 화일관리용 명령을 살펴보면 다음과 같다.

- **DDIR**: 확장된 디렉토리를 나타내며 DOS의 DIR명령과 비슷하다. 이 명령을 사용한 한 예를 화면 4에 나타내었다.
- **DDEXCLUD**: 변형처리 작업으로부터 보호시킨다.

DDEXCLUD는 디렉토리에 보호화일(Exclusion File) "B_._"을 생성시킴으로써 변형처리 작업시 일부 화일 또는 모든 화일을 변형작업에서 제외시킨다.

참고로 화면 5는 이 기성씨의 '알기 쉬운 dBASE'에 있는 실제 화일을 압축시킨 결과이다.

〔화면 5〕 **dBASE** 화일을 압축시킨 결과

```
C>DDIR C:\LEE
 DD RESIDENT IN MEMORY

  Disk Doubler DIRECTORY -- DDIR
  Directory of C:\LEE

               Uncompressed  Compressed  Compression  Date      Time
Filename.EXT   File  Size    File  Size   Ratio       00-00-00  00:00a
------------   ------------  ----------  -----------  --------  ------
BOOK    .DBF           1749         448         3.90  10-10-88   1:42p
BOOK1   .NDX           1025         332         3.09  10-10-88   1:42p
COMPANY .DAT            896         167         5.39  10-23-86   1:57p
CONFIG  .SYS             22          22         1.00  12-30-87   4:38p
DBDISK-1.SUL           6784        5670         1.20   4-19-88   8:41a
DISK-B                 2846        2452         1.16   4-18-88   3:30p
FILE0                   384          70         5.49   9-22-86   6:27p
FSCARGO .FRM           2048        1224         1.67   6-28-88  12:12a
JOURNAL .DBF    [illegible]         857         2.32   4-18-88  10:20a
[illegible]                                     3.17   4-18-88  10:20[illegible]
[illegible] .HLP        795         795
KG      .HLP            506         340         1.49  [illegible] [illegible]54a
KYAD    .DBF           4096        1855         2.21   6-28-88  12:15a
MASTER                28000        2896         9.67   1-01-80  12:22a
SKG6    .DBF           3584        1803         1.99   6-28-88  12:14a
TEETH   .DBF          12093        2153         5.62   1-03-80  12:09a
TRANS                 20480       18039         1.14   1-01-80  12:18a
------------   ------------  ----------  -----------
Files   25           100902       47030    Avg.  2.15

          Total Bytes available on disk: 18682096
```

비디오램의 속성변환

최명환

Edix는 워드스타보다 프로그램의 길이가 작고, 에디터만을 위한 펑션키가 잘 배치되어 있고 무엇보다도 모든 데이터를 RAM에 넣고 처리하기 때문에 처음 라인부터 끝 라인까지 신속하게 이동을 할 수 있다.

반면 워드스타는 디스크와 RAM의 데이터를 번갈아 가면서 편집하기 때문에 화일이 큰 경우 처음부터 끝까지 가자면 화일의 모든 데이터를 디스크로 부터 읽어야 한다. 그러므로 램 디스크상에서 화일을 신속히 편집한후 컴파일하고 다시 에러를 고치자면 좀 답답한 느낌이 든다. 그러나 워드스타에서 Edix로 바꾸고 나서 좀 문제가 생겼다.

필자는 가끔 삼성컴퓨터를 쓸때가 있는데 여기에 Edix를 띄우면 화면이 반전되어서 모니터를 보기 어렵다. 이것은 컴퓨터가 EGA보드를 내장하고 있기 때문에 VRAM의 속성(Attribute)이 07H로 표시되는 다른 컴퓨터와는 달리 17H로 채우기 때문이라고 생각된다. 그래서 모니터의 Bright 볼륨을 조정해서 좀 어둡게 사용하면 반전이 없어지나 에딕스를 빠져 나오게 되면 글자가 보이지 않게 된다. 그러면 다시 Bright의 볼륨을 높여야 한다. 그런데 이것은 에디트와 컴파일을 번갈아 할때마다 계속 볼륨을 조정해야 하기 때문에 여간 귀찮은 것이 아니다.

비디오램의 속성변환

그래서 이 반전을 어떻게 하면 없앨 수 있을까 하고 생각한 끝에 착안한 것이 여기에 소개하는 프로그램이다.

원리는 매우 간단하다. 화면에서 비디오램의 속성값 17H를 모두 정상모드인 07H로 바꾸어 주는 것이다.

독자 여러분들도 잘 아시다시피 IBM PC는 매초마다 18.2번의 타임 인터럽트를 실행한다. 이때 타임 인터럽트는 8H이다. 그러므로 매번 INT8H를 실행할 때마다 가로채어서 비디오램을 읽고 속성을 07H로 바꾸어 주면 화면의 반전이 없어지게 된다. 그러나 매번 INT8H를 실시할 때마다 2000바이트나 되는 화면 데이터를 바꾸자면 전체적으로 볼때 PC의 속도가 떨어지게 된다. 이것을 INT8H가 10번 발생할때 한번정도 비디오램을 고치거나 5번에 한번 정도만 바꾼다면 PC의 수행능력에는 큰 차이가 없을 것이다.

이것을 조정하는 것이 다음 프로그램 리스트의 레이블 V-count이다. V-count를 1로 하면 매번 INT8H를 할때마다 비디오램을 고치게 되고 7로 하면 7번에 한번꼴로 비디오램을 고치게 된다.

여기서 눈치 빠른 독자라면 화면의 속성 뿐만 아니라 화면의 문자까지 바꿀 수 있다는 것을 알게 될 것이다.

가령 화면의 'A'라는 문자나 세미그래픽을 어느 특정한 값 즉, 'P'나 '#'로 바꿀 수 있다. 만일 화면에 표시되는 'A'라는 모든 문자를 'C'로 바꾸고 싶다면

```
CHAR  DB  'A','C', 0FFH, 0FFH
```

로 고치면 된다. 여기서 16진수 값의 0FFH, 0FFH는 여기가 마지막이라는 인식 숫자이다.

또 화면에 역상문자로 표시되는 부분을 하일라이트 문자로 바꾸고 싶을 때는

```
ATTR  DB  112, 16, 0FFH, 0FFH
```

로 고치면 된다.

그밖에 생각하기에 따라서 화면위에 하일라이트 부분을 깜박거리게 하거나 깜박거리는 문자를 역상문자로 바꿀 수가 있다. 참고로 모노크롬인 경우 대부분 프로그램의 속성은 하일라이트일 때 16(0FH), 역상은 112(70H), 블랭킹(blinking)은 230(F0H)이다.

여러분이 칼라보드를 갖고 있는 경우, 프로그램을 만든 회사들이 화면을 멋있게 보이게 하기 위하여 울긋불긋하게 치장하여 놓기 때문에 속성이 모노크롬에서 실행시킬 때와 칼라보드에서 실행시킬 때와는 많은 차이가 있다.

필자도 PC TOOLS의 화면 속성을 조사해 보니, 하늘색(30H), 빨강색(40H), 분홍색(50H), 연녹색(2FH), 그외에도 무수하게 많이 바뀐 것을 알 수 있었다(EGA 보드인 경우).

램상주시키는 법

다음에 소개하는 프로그램은 물론 램상주 프로그램이다(리스트 1 참조). 그러나 지금까지 소개된 램상주 프로그램과 설치하는 방법이 좀 다르다. 기존의 램상주 프로그램은 원하는 인터럽트 번호의 벡터를 프로그램 벡터로 바꾸어 주고 프

〔리스트 1〕 램상주 프로그램

```
           vector8h=4*8h
           vector80h=4*80h
stack  segment  stack 'stack'
       db      211  dup(0)
stack  ends
code  segment  'code'
           assume   cs:code,ds:code,ss:stack
entry      proc     far
           mov      ax,cs
           mov      ds,ax
           mov      es,ax

           mov      ah,35h          ┐
           mov      al,8h           │ INT 8H의 벡터를 읽는다.
           int      21h ;es:bx      ┘
           mov      cx,es
           mov      ax,00h
           mov      es,ax
           mov      di,vector80h;=4*80h           ┐
           mov      word ptr es:[di],bx           │ INT 80H의
           mov      word ptr es:[di+2],cx         │ 벡터를 카피한다.
           mov      word ptr cs:[int8h],bx        │
           mov      word ptr cs:[int8h+2],cx      ┘

           mov      di,vector8h;=4*8h             ┐
           lea      ax,main                       │ 프로그램의 벡터를
           mov      word ptr es:[di],ax           │ INT 8H에 카피한다.
           mov      word ptr es:[di+2],cs         ┘
rdos:      lea      dx,space1       ┐
           int      27h             ┘ 램을 상주시키는 부분
save_reg   proc
           mov      cs:word ptr [ax_reg],ax       ┐
           mov      cs:word ptr [bx_reg],bx       │
           mov      cs:word ptr [cx_reg],cx       │
           mov      cs:word ptr [dx_reg],dx       │ 모든 레지스터를
           mov      cs:word ptr [si_reg],si       │ 세이브
           mov      cs:word ptr [di_reg],di       │
           mov      cs:word ptr [ds_reg],ds       │
           mov      cs:word ptr [es_reg],es       ┘
           ret
save_reg   endp
load_reg   proc
           mov      ax,cs:word ptr [ax_reg]       ┐
           mov      bx,cs:word ptr [bx_reg]       │
           mov      cx,cs:word ptr [cx_reg]       │
           mov      dx,cs:word ptr [dx_reg]       │ 모든 레지스터 복귀
           mov      si,cs:word ptr [si_reg]       │
           mov      di,cs:word ptr [di_reg]       │
           mov      ds,cs:word ptr [ds_reg]       │
           mov      es,cs:word ptr [es_reg]       ┘
           ret
load_reg   endp
int8       proc
           int      80h
           cli
           call     save_reg
           sti
           pushf
           cli
           pop      ax;flag         ┐
           pop      bx;call saved   │
           pop      cx;ip           │
           pop      dx;cs           │
           pop      di              │
           push     ax              │ INT8H 실행부분
           push     dx              │
           push     cx              │
           push     bx              │
           ret                      │
int8       endp                     ┘
convert    proc
           mov      ax,cs                           ┐
           mov      ds,ax                           │
           dec      cs:byte ptr [v_count1]          │
           jz       cvt_3                           │
           ret                                      │
cvt_3:     mov      al,cs:byte ptr [v_count]        │
           mov      cs:byte ptr [v_count1],al       │
           call     mono_color                      │
           mov      ax,cs:word ptr [video_adr]      │
           mov      es,ax                           │ 비디오램의
           mov      di,00h                          │ 데이터를
           mov      cx,2000                         │ 바꿔주는 부분
cvt_4:                                              │
           mov      ax,es:[di]                      │
           call     attr_search                     │
           call     char_search                     │
           mov      es:[di],ax                      │
           inc      di                              │
           inc      di                              │
           loop     cvt_4                           ┘
           ret
video_adr  dw       0000h
v_count    db       7          120의 속성을 10으로 바꾼다.
v_count1   db       7          (삼성의 경우 17H를 07로)
attr       db       120,10,0ffh,0ffh,50 dup (0ffh)
char       db       'a','A',0ffh,0ffh,50 dup (0ffh)
convert    endp                'a'의 모든 문자를 'A'로 바꾼다.
attr_search proc
           lea      bx,attr-2       ┐
a_s_5:     inc      bx              │
           inc      bx              │
           mov      dx,[bx]         │ 레이블 attr로 부터 변환하고자
           cmp      dx,0ffffh       │ 하는 코드를 읽어 변환
```

```
                jz      a_s_6
                cmp     dl,ah ;attr
                jnz     a_s_5
                mov     ah,dh
a_s_6:          ret
                ret
attr_search     endp
char_search     proc
                lea     bx,char-2
c_s_5:          inc     bx
                inc     bx
                mov     dx,[bx]           레이블 char로 부터
                cmp     dx,0ffffh         변환하고자 하는 코드를 읽어 변환
                jz      a_s_6
                cmp     dl,al ;char
                jnz     a_s_5
                mov     al,dh
c_s_6:          ret
                ret
char_search     endp
mono_color      proc
                mov     ax,00h
                mov     es,ax
                mov     di,449h
                cmp     es:byte ptr [di],07h
                jz      mono_color_5
                mov     cs:word ptr [video_adr],0b800h
                ret
mono_color_5:   ;it is a mono board
                mov     cs:word ptr [video_adr],0b000h
                ret
mono_color      endp
main            proc
                call    int8              랩상주의 실행부분
                call    convert
                call    load_reg
                iret
main            endp
ax_reg          dw 00h
bx_reg          dw 00h
cx_reg          dw 00h
dx_reg          dw 00h
si_reg          dw 00h
di_reg          dw 00h
ds_reg          dw 00h
es_reg          dw 00h
int8h           dw      0,0
space           db      1000 dup('0')
space1          db      ' '
entry           endp
;-----------------------------------
                code            ends
                end             entry
```

(mono_color 부분 주석: 갖고 있는 보드가 모노인지 칼라보드인지 체크하여 모노일 경우 비디오번지를 0B000H로, 칼라일 경우 비디오번지를 0B800H로 바꾼다.)

로그램내에서 Long 점프나 Call로 원래의 인터럽트를 수행했으나, 이 프로그램에서는 원하는 인터럽트 벡터를 새로운 인터럽트 벡터로 복사한다(여기서는 80H 이후). 그리고 프로그램내에서 인터럽트를 실행하고 IRET 한다(예를 들면 INT80H).

그 외에도 램상주를 하는 방법은 여러가지가 있는 줄로 안다. 독자 여러분들도 그 가능성에 대해 생각해 보기 바란다.

램상주의 제거방법

다음은 램상주 제거에 대해서 생각해 보자. 필자는 여기에 대해서 두가지 방법을 쓰고 있다.

첫번째 방법은 램상주 프로그램내에 램제거 루틴을 넣는 것이다. 가령

```
A>VR *
```

이라고 입력하면 제거루틴은 PSP의 DTA내에서 *표를 인식하고 원래의 인터럽트 벡터로 환원시킨다. 그러나 이렇게 하자면 프로그램은 매우 복잡해진다. 램상주를 가동하기 전에 인터럽트 벡터를 세이브해두고, 램상주가 있나 없나 체크를 해야 되며 PSP의 처리루틴을 넣어야 한다. 그래서 필자는 좀 조잡하기는 하지만 누구나 쉽게할 수 있는 두번째 방법을 소개한다.

프로그램 작성방법

이것은 또하나의 프로그램을 만드는 것이다. 이때는DEBUG와 GWBASIC이 필요하다.

다음의 베이직 프로그램을 실행시켜 보자. 그러면 인터럽트 0번부터 차례로 표시될 것이다. 여기서 인터럽트 8H의 8=F000 : FEA5를 보라(필자의 대만제 PC의 경우이다. 그러나 다른 컴퓨터 심지어 AT까지도 벡터가 같은 것을 확인했음).

```
10 DEF SEG=0:X=0:I=0
20 PRINT HEX$(I);"=";HEX$(PEEK(X+3));" ";HEX$(PEEK(X+2));":";
25 PRINT HEX$(PEEK(X+1));" ";HEX$(PEEK(X  )),
30 X=X+4:I=I+1
40 GOTO 20
```

물론 이때는 램상주를 올리기전의 원래 고유의 벡터이다. 자, 이것을 노트에 적어 두고 베이직으로 부터 빠져나오고 디버그의 커맨드모드로 들어가자. —A명령으로 다음을 어셈블시킨다.

```
MOV     AX,0000
MOV     ES,AX
ES:
MOV     BYTE PTR [0020],A5
```

```
ES:
MOV     BYTE PTR [0021],FE
ES:
MOV     BYTE PTR [0022],00
ES:
MOV     BYTE PTR [0023],F0
MOV     AH,4C
INT     21
```

다음에 이것을 VRET. COM이라는 화일로 세이브 해보자. 먼저 -nVRET. COM으로 화일이름을 정해준 후 -Rbx에서 bx 값을 0000로 준다. 그리고 -Rcx에서 cx값은 넉넉하게 100 정도로 설정한다. 여기서 cx레지스터는 프로그램의 길이를 나타낸다.

다음에 -W 하고 리턴키를 쳐라. 그러면 Writing 0100bytes 라는 메시지를 표시하고 다시 코맨드 모드로 돌아올 것이다. 그런 다음 디버그를 빠져 나와

```
A>DIR VRET.COM
```

으로 확인해 보라. 만약 틀림없이 입력했다면 데이터들이 바뀌었을 것이다. 램상주를 올린 후 필요할 때 VRET. COM을 작동시키면 램상주는 제거될 것이다. 이러한 방법은 좀 무식한 방법이 될것 같으나 어셈블러 프로그램에 있어서 디버그의 존재는 무시 못한다.

필자는 어셈블러 프로그램시 가장 골이 아픈 로직 에러를 찾고자 할때 종종 디버그를 사용한다.

시스템 자원(2)

DOS는 프로그램이 COMMAND.COM을 메모리에 저장 하게끔 하여 COPY나 DIR명령과 같은 명령들을 사용할 수 있게끔 한다. 실행가능한 화일은 확장자가 두가지 형식 즉 .EXE와 .COM만이 가능하다. EXE프로그램은 프로그램의 길이가 64KB이상이 되어도 괜찮으나, 프로그램을 메모리에 로드할 때 DOS가 어떤 프로세싱(Processing)을 행하도록 한다. 이와는 다르게 COM 프로그램은 메모리가 요구하는 형으로 이미 존재한다.

COM 프로그램은 간단한 용도에 아주 유용하며 어느 경우도 프로그램을 구성하는 코드는 PSP(Program Segment Prefix)에 의해 메모리에 실린다. PSP는 프로그램을 수행하는데 필요한 특수 정보인 DOS가 들어 있는 100H 바이트의 크기를 가지고 있으며 화일 I/O기능을 위해 일정한 공간을 제공한다. EXE 화일이 로드됐을 때 DS와 ES는 PSP를 지적하며 COM화일의 경우는 CS가 초기에 PSP를 지적한다.

DOS는 PSP 세그먼트 번호를 리턴시키는 기능을 갖고 있다. 이것은 INT 21H의 번호 62H로 입력 레지스터는 없으며, 패러그래프(Paragraph) 번호를 BX에 옮기는 역할을 한다.

PSP 위치는 그 첫째 워드(Word)가 프로그램을 종료시키는 DOS 인터럽트의 번호에 관계한다. 프로그램의 최종 RET 문이 실행됐을 때, 스택의 최고점에 있는 값들이 지시 포인터(IP 레지스터)를 PSP의 시작점에 연결시킴으로써 종료 코드는 프로그램의 그 다음 지시에 의해 실행된다.

표 1은 PSP의 영역을 나타낸다.

메모리의 할당과 할당해제

DOS가 프로그램을 로드할 때, 프로그램은 COMMAND.COM과 인스톨된 어떤 디바이스 드라이버나 이미 상주하고 있는 DOS 유틸리티들 바로 위에 있는 메모리의 「최저점」에 놓이게 된다. 이때 프로그램 위에 있는 모든 메모리는 프로그램에 할당된다. 만일 프로그램이 데이터 영역을 확보하기 위해 어떤 메모리가 필요하다면 메모리상에 프로그램 코드가 끝나는 지점을 파악한 다음 그 다음의 어떤 지점이라도 데이터 영역을 자유롭게 구성할 수 있다.

프로그램이 끝나는 지점을 조사하기 위하여 다음과 같이 프로그램 끝에다 "의사 세그먼트(Pseudo Segment)"를 놓는다.

```
ZSEG     SEGMENT
           .
           .
ZSEG     ENDS
```

IBM 어셈블러에서는 세그먼트들이 알파벳 순서에 의해서 배열되기 때문에 ZSEG는 맨끝에 위치하게 된다. 다른 어셈블러에서도 소스코드의 맨끝에 위치한다. 프로그램에서 간단히 MOV AX, ZSEG라고 쓰게 되면 AX는 프로그램에 따르는 메모리내 최초의 비어있는 세그먼트를 나타낸다.

이러한 액세스방법은 프로그램이 실제로 없는 메모리에 존재한다고 가정하지 않는한 사용이 되지만, 여러개의 프로그램이 같은 범위의 메모리 어드레스를 사용하는 다수 사용자 환경하에서는 적용 할 수 없다. 이런 문제에

〔표 1〕 PSP의 영역

오프셋	영역의 크기	사 용
0H	DW	DOS기능의 번호는 프로그램을 종료시키는데 사용
2H	DW	패러그래프의 메모리 크기
4H	DW	보류
6H	DD	DOS기능 발송자에 대한 롱콜(long call)
AH	DD	종료 어드레스(IP, CS)
EH	DD	제어-브레이크(Ctrl-Break) 탈출(exit) 어드레스(IP, CS)
12H	DD	중대한 에러 탈출 어드레스(IP, CX)
16H	22바이트	보류
2CH	DW	프로그램의 주변 스트링의 패러그래프 번호
2EH	46바이트	보류
5CH	16바이트	파라미터 영역 1
6CH	20바이트	파라미터 영역 2
80H	128바이트	결점 디스크 전달영역/명령라인 데이터 인수

대한 해결책으로써 DOS는 시스템안에 640K의 메모리 트랙을 유지하여야 한다.

메모리 블록은 단순히 연속하는 메모리 섹션이지만, 메모리가 허용되는 만큼 늘릴 수도 있으며, 특히 1 세그먼트(64K)보다 크게 할 수 있다. 만일 너무 큰 블록을 요구한다면 DOS는 에러 메시지를 나타낸다.

그리고, 어떤 경우라도 블록을 중복시킬 수는 없으며 DOS가 존재하는 블록을 할당해제하거나 혹은 확장시키거나 줄일 수는 있다.

어떤 메모리 할당이 구성되기 전에 존재하는 블록(프로그램의 시작점에서부터의 모든 메모리)은 프로그램 크기로 줄여야 한다. 그 다음 한 블록이 생성되면 DOS는 16바이트 메모리 제어블록(Memory Control Block)을 구성하여 메모리안에서 그 블록보다 앞서도록 한다.

표 2에서 보는 바와 같이 최초의 5바이트는 다음과 같은 기능을 한다.

DOS는 블록을 한 체인(Chain)으로 받아들이며 처음 블록의 어드레스는 내부 변수로 유지된다. DOS는 그 변수에 의해 처음에 할당된 블록의 메모리 제어블록을 배치한 다음 포함된 정보로부터 다음 블록을 **그림 1**과 같이 찾아 나간다.

사용자는 한번 DOS 메모리 할당 시스템을 시작하면 이것을 그대로 유지해야 한다. 만일 프로그램이 계속하여 제어 블록을 포함하는 영역을 과다하게 사용하면, 체인은 끊어지고 DOS는 에러메시지를 내보낸다.

DOS는 INT 21H의 번호 48H~4AH인 3개의 메모리할당 기능을 제공한다.

세번째 기능("SETBLOCK")은 현재 프로그램에 할당된 메모리 크기를 변화시키는데, 다른 기능이 사용되기 전에 실행되도록 하는 기능이다. 그리고 블록들은 자유롭게 할당되거나 할당해제될 수 있다.

프로그램은 종료되기 전에 할당된

〔리스트 1〕

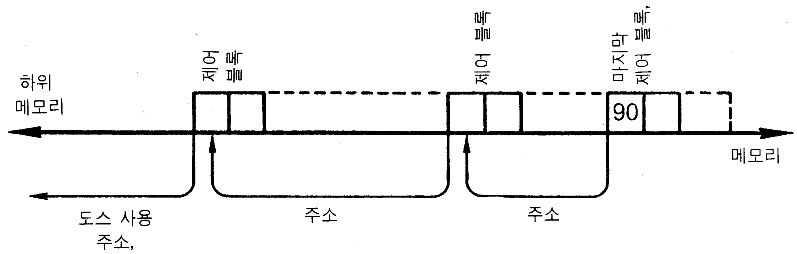
```
;--1024바이트 영역 할당
    MOV   AH,48H        ;기능 번호 세트
    MOV   BX,64         ;16바이트 패러그래프 64개를 세트
    INT   21H           ;할당
    JC    ERROR         ;carry가 발생하면 에러루틴으로 점프
    MOV   BLOCK_SEG,AX ;아니면, 블록 세그먼트 저장

;--동일 영역의 할당 해제
    MOV   AX,BLOCK_SEG  ;블록의 시작번지 지정
    MOV   ES,AX         ;AX 즉 시작번지를 ES로 세트
    MOV   AH,49H        ;해제 기능번호
    INT   21H           ;실행
```

〔표 2〕

BYTE 0	체인의 마지막 블록이면 아스키 코드 90, 아니면 아스키 코드 77이다.
BYTES 1~2	메모리 할당해제가 되면 0으로 된다.
BYTES 3~4	패러그래프안의 블록을 중단시킨다.

〔그림 1〕 메모리 제어 블록



모든 블록들을 할당해제시켜야 한다. 그렇지 않으면 메모리는 다음에 실행되는 프로그램에 속해 버린다.

●어셈블리어의 경우

INT 21H 3개의 메모리할당 기능 모두가 운영하는 메모리 블록을 위해 16비트 시작 어드레스를 사용한다. 이 어드레스는 블록이 시작되는 세그먼트를 나타낸다.(블록은 항상 그 세그먼트안의 오프셋(Offset) 0에서 시작한다.) 그래서 실제 블록 시작배치는 이 값에다 16을 곱한 메모리 어드레스에서 행한다. 또한, 세가지 기능 모두에서 BX는 할당되었거나 비어있는 16바이트 메모리 섹션(Paragraph)의 번호를 지닌다. 만일 기능이 실패하면 캐리 플래그(Carry Flag)를 세트하고 AX는 그 이유를 설명하는 에러코드를 내보낸다. 적합한 3개의 코드는 **표 3**과 같다.

할당 기능은 코드 7과 8을 사용하고, 할당해제 기능은 코드 7과 9를 사용하는데, 할당을 변경시키는 기능은 이 3개의 코드를 모두 사용한다. 그 다음 코드는 처음 1024바이트 블록을 할당하여 변수 BLOCK_SEG.BX에 시작어드레스를 저장하여, 요구하는 16바이트 패러그래프의 번호를 갖게 되어 AX : 0에 의해 알 수 있는 시작어드레스로 옮긴다. 코드의 제2 부분은 프로

그램이 종료됐을 때 요구하는 것과 똑같은 블록을 할당해제한다. 이 경우, AX로 옮겨진 값은 ES에 놓인다. DOS는 블록 크기를 추적하여 얼마나 많은 패러그래프가 할당해제되는지 알아낸다(리스트 1 참조).

마지막으로 기능 4AH에 대한 예를 보자. ES는 PSP 즉, 프로그램이 실린 메모리안의 최초 바이트의 세그먼트 값을 갖는다. ES는 시작점에서 이 값으로 초기화한다. SETBLOCK를 사용하기 위해 프로그램의 시작점에서 기능을 호출하거나 나중 사용을 위해 ES의 초기값을 저장한다.

BX는 16바이트 패러그래프에 원하는 블록 크기를 갖는다. 크기를 계산하기 위해 프로그램에 소스 코드의 끝에 위치하는 여분의 가상 세그먼트를 넣는다. IBM 매크로 어셈블러에서는 세그먼트는 알파벳 순서에 의해 놓이고, 이 가상 세그먼트는 "ZSEG"와 같은 이름을 갖게 되면 소스 코드의 어느 위치에 놓여도 된다. 다른 어셈블러에서는 그 가상 세그먼트를 실제 소스 코드의 끝에 넣는다. 프로그램이 요구하는 메모리가 제공된다면, 프로그램은 이 세그먼트를 읽어 들여 그 자신의 시작 세그먼트와 비교한다. 프로그램이 실렸을 때 ES와 DS 모두 PSP에서 프로그램이 시작되는 패러그래프 번호를 갖는다(리스트 2 참조).

다른 프로그램에서 프로그램의 실행

DOS에서 한 프로그램을 다른 프로그램 안에서 실행시키기 위해서는 EXEC 기능 (INT 21H의 번호 4B)을 사용한다. 제1 프로그램을 "메인"이라 부르고, 그 안에 실려 실행되는 프로그램을 "서브"라 부른다.

●어셈블리어의 경우

기능 4BH는 사용이 복잡하며 다음 4가지 준비 단계를 필요로 한다.

① 프로그램을 위해 메모리에서 가능한 공간을 확보한다.
② 파라미터에 블록을 설정한다.
③ 드라이브, 패스(Path), 프로그램 이름 스트링을 구성한다.
④ SS와 SP 레지스터를 변수로 저장한다.

프로그램이 실리면 DOS는 전체 메모리를 할당하기 때문에 메모리안에 공간을 확보해야 한다. 메모리가 비어 있지 않으면 다음 프로그램을 실을 수 없다. 일단 메모리가 비어 있으면, BX에 메모리 공간의 16바이트 패러그래프의 번호를, AH에 4AH를 넣은 다음 요구된 패러그래프의 번호를 프로그램에 적합하도록 메모리 할당을 줄이기 위해 INT 21H를 실행한다.

ES:BX가 나타내는 파라미터 블록(Parameter Block)은 14바이트 블록의 메모리로 표 4와 같은 4가지의 정보로 나타난다.

인바이런먼트 스트링(Environment String)은 프로그램을 실행시킬 때 DOS가 지켜야 할 하나 또는 그 이상의 규정에 대한 스트링이다. 인바이런먼트 스트링은 디스크내의 CONFIG.SYS 화일에 나타나는 것과 똑같다. 예를 들어 VERIFY=ON은 스트링으로 놓을 수 있다. 최초 요소(Element)는 스트링으로, 최종 요소는 ASCII 0 캐릭터로 시작하여 계속 써내려 간다. 최종 요소는 2개의 ASCII 0캐릭터를 수반해야 한다. 스트링은 패러그래프 경계에서 시작한다. 왜냐하면, 스트링을 나타내는 파라미터 블록의 엔트리는 2바이트 세그먼트 값을 갖고 있기 때문이다. 이것은 새로운 프로그램이 실리기 위한 것과 똑같은 인바이런먼트 스트링으로 운영된다면 피할 수 있다. 이 경우 최초 2바이트의 파라미터 블록에 간단히 ASCII 0을 넣으면 된다.

파라미터 블록의 그 다음 4바이트는 프로그램이 실릴 때 명령 라인을 나타내기 위한 것이다. 명령라인은 프로그램을 불러내기 위한 스트링이다.

모든 레지스터는 호출하는 프로그램에 따라 바뀐다. 스택 세그먼트와 스택 포인트는 저장되었다가 제어가 프로그램 호출로 옮길 때 복원된다. 이렇게 하여 변수를 따로 세트 시킨다. DS는

〔표 3〕

7	메모리 제어 블록은 파괴된다.
8	불충분한 가용메모리이다.
9	메모리 블록 주소가 등록되어 있다.

〔표 4〕

DW	인바이런먼트 스트링의 세그먼트 주소
DD	코맨드 라인의 세그먼트 옵셋
DD	첫번째 화일 제어 블록 세그먼트 옵셋
DD	두번째 화일 제어 블록 세그먼트 옵셋

〔리스트 2〕

```
;프로그램 메모리의 재할당
    MOV   BX,ZSEG       ;paragraph번호(프로그램의 하단 +1)
    MOV   AX,ES         ;paragraph번호(프로그램의 시작번지)
    SUB   BX,AX         ;프로그램의 사이즈 계산
    MOV   AH,4AH        ;기능번호
    INT   21H           ;재할당작업 수행
    JC    MEMORY_EROOR  ;에러 체크



;----더미 세그먼트
 ZSEG         SEGMENT
 ZSEG         ENDS
```

소멸됐기 때문에 이러한 변수들은 세그먼트 MOV AX, DSEG와 MOV DS, AX가 반복될 때까지 복원되지 않는다. 한편, SS와 SP가 저장되어 있기 때문에 로드 및 실행 옵션을 선택하기 위해 AL에 0을 넣는다. 그 다음 AH에 4AH를 넣고 INT 21H를 호출한다.

이러한 점에서 2개의 프로그램은 효율적으로 실행되고, 메인 프로그램은 일단 "정지"한다. DOS는 메인 프로그램에 리턴 코드(Return Code)를 옮기는 방법을 서브 프로그램에 제공하여 에러와 상태를 알 수 있도록 한다.

에러가 발생하면 캐리 플래그가 세트되고, 이 경우 AX는 기능번호가 맞지 않을 때는 1로, 화일을 찾을 수 없을 때는 2로, 디스크 문제일 때는 5로, 불충분한 메모리일 때는 8로, 인바이런먼트 스트링이 맞지 않을 때는 10으로, 포맷이 맞지 않을 때는 11로 옮긴다.

전체 파라미터 블록이 0일때는 인바이런먼트 스트링을 구성하지 않는다. 이것은 명령 라인이 실려 있는 프로그램을 통과하지 않고, 인바이런먼트를 호출 프로그램과 똑같이 취급한다는 뜻이다. 메모리 할당을 변경시키려면 화일명과 (비어있는) 파라미터 블록을 구성하고 SS와 SP를 저장한다(리스트 3 참조).

〔리스트 3〕

```
;--- 데이터 세그먼트
 FILENAME      DB   'A:TRIAL.EXE',0  ;A드라이브로 부터 TRIAL.EXE를 로드
 PARAMETERS    DW   7 dup(0)         ;파라미터 블록을 모두 0로 채움
  KEEP_SS      DW   0                ;SS의 변수
  KEEP_SP      DW   0                ;SP의 변수

;---메모리의 재할당
   MOV   BX,ZSEG        ;프로그램 끝의 paragraph번지
   MOV   AX,ES          ;프로그램 시작의 paragrapf번지
   SUB   BX,AX          ;프로그램 사이즈 계산
   MOV   AH,4AH         ;기능 번호
   INT   21H            ;재할당 실행
;---파라미터 블럭의 위치
   MOV   AX,SEG PARAMETERS    ;ES에 세그먼트 설정
   MOV   ES,AX                ;
   MOV   BX,OFFSET PARAMETERS ;BX에 오프셋 설정

;---SS와 SP의 저장
   MOV   KEEP_SS,SS      ;SS저장
   MOV   KEEP_SP,SP      ;SP저장

;---화일이름 스트링의 위치
   MOV   DX,OFFSET FILENAME.  ;DX에 오프셋값 지정
   MOV   AX,SEG FILENAME      ;DS에는 세그먼트값 지정
   MOV   DS,AX

;---프로그램 로드
   MOV   AH,4BH               ;EXEC 기능
   MOV   AL,0                 ;"로드와 실행" 선택
   INT   21H                  ;실행

;---레지스터의 재저장
   MOV   AX,DSEG              ;DS저장
   MOV   DS,AX                ;
   MOV   SS,KEEP_SS           ;SS저장
   MOV   SP,KEEP_SP           ;SP저장
;---프로그램 종료시 이를 알리기 위한 더미 세그먼트 생성
   ZSEG          SEGMENT
   ZSEG          ENDS
```

DOS유저 인터페이스 명령의 사용

프로그램은 DIR이나 CHKDSK와 같은 DOS사용자 인터페이스 명령을 전 범위에 걸쳐 마음대로 가질 수 있다. 이러한 서비스가 프로그램내로부터 사용되면 COMMAND.COM의 제2 카피가 실려 수행된다. 많은 양의 프로그램이 이러한 방법으로 저장되면 이 제2 카피에 대한 적당한 메모리를 할당해야 하는데, 이 메모리가 충분하지 않으면 프로그램은 통과하지 않고, 그대로 남아있게 된다.

베이직은 SHELL문을 사용하여 COMMAND. COM의 제2 카피를 로드한다.

COMMAND.COM은 화일명이 규정되기 전에 실리며 간단히 SHELL을 써서 DOS로 옮긴다.

DOS유틸리티중에 어느 하나가 사용되면 배치화일을 포함한다. 이는 호출 베이직 프로그램으로 리턴하기 위해서 EXIT로 들어간다.

●어셈블리어의 경우

보통 명령라인은 길이를 나타내는 바이트로 시작하고 명령 스트링은 그 자체이며, ASCII 13으로 끝난다.

COMMAND.COM으로 명령을 옮겼을 때 스트링전에 /C를 넣어야 한

다. COMMAND.COM이 있는 드라이브를 규정짓고 명령스트링의 시작점에 드라이브 프레픽스(Drive Prefix)를 놓아야 한다. COMMAND.COM이 드라이브 B에 들어있을 때, 드라이브 A의 디렉토리를 갖기 위해서는 다음과 같이 쓴다.

```
COMMAND_LINE  DB 12,'B:/C DIR A:',13
```

다음의 비트 코드는 파라미터 블록에 명령 라인 어드레스를 세트시킨다(리스트 4 참조).

프로그램을 메모리에 상주시키는 방법

메모리에 남겨진 프로그램은 다른 프로그램에 여러가지 이익을 준다. 보통 그와 같은 프로그램은 비사용(Unused) 인터럽트 벡터를 통하여 액세스된다. DOS는 프로그램 자체를 한 부분으로 생각하여 다음에 실리는 프로그램에 의해 겹치지 않도록 보호한다. 상주하는 프로그램은 보통 COM형태로 쓰인다.

INT 27H로 끝내는 프로그램은 상주하게 되는데 INT 27H가 실행될 때 CS는 이러한 기능을 적당히 수행시키기 위해 프로그램 세그먼트 프레픽스의 시작을 알려 준다. COM프로그램에서 CS는 초기에 이 위치에 세트되어 프로그램을 27H로 간단히 끝낼 수 있다. 이와는 다르게 EXE 프로그램에서 CS는 초기에 PSP에서 규정한 최초 바이트(즉, 100H)를 나타낸다. 그리고 EXE 프로그램 종료시 최종 RET지시는 최초의 값을 스택에 밀어 넣는다. 즉, PUSH DX / MOV AX, 0 / PUSH AX / DS는 최초 PSP의 맨 끝을 나타내고, 이러한 값들은 지시 포인터에서 나와 INT 20H에 대한 코드를 갖도록 초기화시킨 PSP에서 오프셋 0에 연결된다. 그 다음 INT 20H가 실행이 되고, 이것은 프로그램을 종료시켜 제어를 DOS으로 옮기는 기능을 가진다.

〔리스트 4〕

```
LEA   BX,PARAMETERS              ;파라미터 블럭의 오프셋
MOV   AX,OFFSET COMMAND_LINE     ;코멘드 라인의 오프셋
MOV   [BX]+2,AX                  ;첫 2바이트 위치지정
MOV   AX,SEG COMMAND LINE        ;코멘드 라인의 오프셋
MOV   [BX]+4,AX                  ;두번째 2바이트 위치 지정
```

〔그림 2〕 EXE 프로그램의 종료

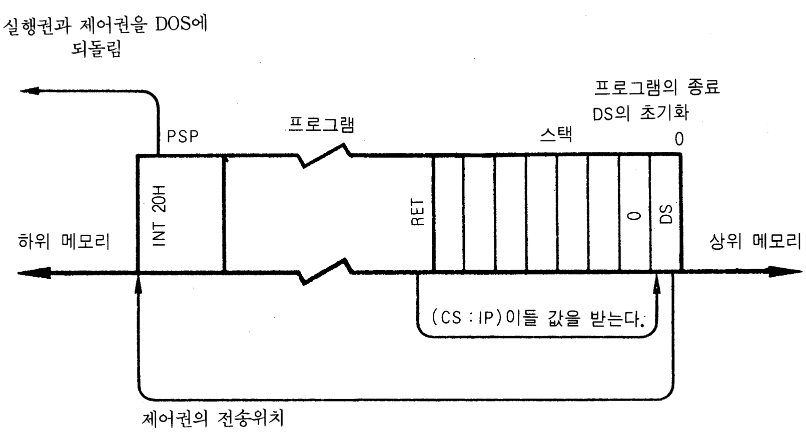


그림 2는 이와 같은 과정을 보여준다. EXE 프로그램에서 INT 27H를 수행하기 위해 PSP의 제 2 바이트에서 27H를 추출한 다음(제1 바이트는 "INT"에 대한 기계 코드를 갖는다.) 통상 RET로 프로그램을 끝낸다. 두 종류의 화일에 있어서 INT 27H가 수행되기 전에 DX는 PSP의 시작부터 출발하여 프로그램 종료의 오프셋을 가져야 한다.

● 어셈블리어의 경우

인터럽트 벡터는 INT 21H의 기능 25H를 사용하여 구성된다. IRET로 루틴을 끝내는 것에 유의하라. 셋업 프로그램은 루틴을 제공하는 이외에 인터럽트 벡터 초기화를 수행하고, DX에 인터럽트 루틴의 끝과 종료를 알려준다.

COM 화일에서는 프로그램 끝에 INT 27H를 넣는다. EXE 화일에서는 PSP의 최초 워드로부터 추출한 RET문을 사용하여 프로그램을 끝낸다. 그 다음 연속적으로 실리는 프로그램이 INT 70H를 호출할 때마다 이 과정을 반복 수행한다.

리스트 5에 COM 및 EXE화일에 대한 예를 들었다. 두 경우 모두 인터럽트 루틴의 끝을 나타내기 위하여 "FINISH" 라는 것을 셋업(Set Up)시켰다. COM화일에서 FINISH는 INT 27H에 의해 요구되는 PSP의 시작으로부터 오프셋을 제공한다. EXE화일에서는 오프셋은 PSP를 따르는 최초 바이트로부터 나오며 이 값에다 100H를 더한 오프셋은 PSP의 밑에서 시작하게끔 한다. 프로그램에 루틴을 위치시켜 셋업(Set-up) 코드는 상주부분(Resident Portion)에서 떨어져 나간다. 다른 기법은 MOVSB를 사용하여 루틴에 대한 코드를 오프셋 60H에서 시작하여 160바이트의 메모리를 비운 PSP의 비사용 부분으로 옮긴다.

INT 21H의 기능 31H은 루틴에 의해 요구되는 16바이트 패러그래프의 번호를 DX에 주는 것을 제외하고는 같은 방법으로 수행한다.

이러한 과정의 장점은 Exit Code를 메인 프로그램에 보내 루틴의 상태에 대한 정보를 제공한다는 것이다. 메인 프로그램은 INT 21H의 기능을 통하여 코드를 감지한다.

프로그램 오버레이의 로드 및 실행

오버레이(Overlay)란 프로그램의 본체는 메모리에 존재하면서 디스크에 남아있는 프로그램의 부분을 말한다. 특수한 오버레이의 기능이 요구될 때 오버레이는 메모리에 실리고 프로그램은 이것을 프로시듀어로서 호출한다. 다른 오버레이는 이전 코드와 중복되면서 똑같은 장소의 메모리에 연속하여 로드된다. 예를 들면 데이터베이스 프로그램은 소트루틴(Sort Routine)을 로드한 다음 레포트 제너레이션 루틴(Report Generation Routine)과 중복시킨다. 이러한 기법은 메모리를 보존하기 위해 사용하지만, 일정하게 사용되지 않는 프로시듀어를 사용할 수 있거나 그렇지 않으면 프로그램 운영을 느리게 만드는 디스크 운영에도 사용된다.

●어셈블리어의 경우

DOS는 오버레이를 로드하기 위해 EXEC기능을 사용한다. 이 기능은 INT 21H의 4BH로써 코드번호 0가 AL에 놓일때 한 프로그램이 다른 프로그램 내에서 로드 및 실행할 때도 사용된다. 그러나 AL에 3이 놓이면 오버레이가 로드된다. 이 경우 프로그램 세그먼트 프레픽스는 형성되지 않아 오버레이는 독립 프로그램으로서 셋업(Sep up)되지 않는다. 이 기능은 터닝 콘트롤(Turning Control)이 없으면 단지 오버레이를 로드하기만 한다.

〔리스트 5〕

```
      COM 화일인 경우
;---인터럽트 루틴
    BEGIN:     JMP   SHORT SET_UP   ;short_set_up으로 점프
  ROUTINE      PROC  FAR
               PUSH  DS             ;레지스터 저장
                   .
                   .
                (해당 루틴)
                   .
                   .
               POP DS               ;레지스터 저장
               IRET                 ;인터럽트 루틴
    FINISH     EQU $                ;루틴의 끝 지정
  ROUTINE      ENDP

;---인터럽트 벡터 설정
  SET_UP: MOV  DX,OFFSET ROUTINE ;DX에 루틴의 오프셋값을 저장
          MOV  AL,70H            ;인터럽트 벡터 번호
          MOV  AH,25H            ;벡터를 설정하기 위한 기능
          INT  21H               ;벡터 설정
;---프로그램의 종료
          LEA  DX,FINISH      ;상주루틴의 오프셋값 설정
          INT  27H            ;종료 (루틴은 상주되어 있음)
  EXE 화일인 경우
;---인터럽트 루틴
          JMP    SHORT SET_UP  ;short set_up로 점프
 ROUTINE  PROC   FAR
          PUSH   DS            ;레지스터의 저장
               .
               .
            (해당 루틴)
               .
               .
          POP    DS            ;레지스터 저장
          IRET                 ;리턴
 FINISH:  EQU    $             ;루틴의 끝 지정
ROUTINE   ENDP

;---인터럽트 벡터 설정
  SET_UP: MOV  DX,OFFSET ROUTINE ;DX에 루틴의 오프셋값 기억
          MOV  AX,SEG ROUTINE    ;DS에 루틴의 세그먼트값 기억
          MOV  DS,AX             ;
          MOV  AL,70H            ;인터럽트 벡터 번호
          MOV  AH,25H            ;벡터설정 기능번호
          INT  21H               ;실행
;---프로그램의 종료
          MOV  DX,FINISH+100H       ;상주 루틴의 마지막 오프셋 설정
          MOV  BYTE PTR ES:1,27H    ;PSP에 27H 세트
          RET                       ;종료(루틴은 상주되어 있음)
```

오버레이되는 부분에 메모리를 제공하는 방법에는 2가지가 있다. 메인 프로그램의 본체내에 한 영역을 중복되게 하거나 그렇지 않으면 메인 프로그램의 외부에 메모리를 특별히 할당시키는 방법이 있다.

EXEC기능은 오버레이가 로드되기 위한 장소로서 단지 세그먼트 어드레스(16-바 이트 경계)를 주는 것이다. 오버레이가 한 프로그램에 로드될 때 그 프로그램은 서라운딩코드(Surrounding Code)를 잠식시키는 것을 막아 오버레이를 보호해 주는 패러그래프 번호를 계산해야만 한다. 이와는 다르게 메모리를 분리하여 할당시킬 때 DOS는 패러그래프 번호를 갖는 프로그램을 제공한다.

리스트 6의 예는 메모리 할당방법을 사용한다. DOS는 초기에 모든 가능한 메모리를 프로그램에 할당하기 때문에 최초 기능 4AH는 초과 메모리(Excess Memory)를 할당한다. 기능 48H는 최대 오버레이를 수용할 수 있을 정도의 큰 블록을 할당하여 그 오버레이가 블록에 세트되도록 한다. 이 기능은 블록의 세그먼트 값을 AL에 옮기며, 패러그래프 번호는 오버레이가 로드되는 지점과 메인프로그램에 의해 오버레이가 호출되는 지점이다.

AL에는 코드번호 3을 제외하고, 이러한 기능을 위해 셋업해야 할 다른 2가지의 입력이 있다. DS:DX는 오버레이 화일에 패스(Path)를 제공하면서 ASCII 0바이트로 끝나는 스트링을 나타낸다. DOS는 프로그램 화일을 찾는 것처럼 그 화일을 읽어 들이지 않기 때문에 화일전체의 이름을 준 다음 .COM이나 .EXE로 완성한다.

마지막으로 ES:BX는 4바이트 파라미터 블록을 나타내는데 이 블록은 오버레이가 로드되는 곳의 2바이트 패러그래프 번호와 오버레이내에 재배치(Relocation)하는 어드레스를 위해 사용되는 2바이트 재배치 인자(Relocation Factor)를 포함한다. 패러그래프 번호

〔리스트 6〕

```
;---프로그램의 끝은 메모리 할당을 위해 더미 세그먼트를 취한다.
 ZSEG        SEGMENT
 ZSEG        ENDS

;---데이터 세그먼트:
 OVERLAY_SEG      DW    ?
 OVERLAY_OFFSET   DW    ?
 CODE_SEG         DW    ?    ;오버레이 세그먼트
 PATH             DB    'A:OVERLAY.EXE'
 OBLOCK           DD    0    ;4byte 파라미터 블럭

;---프리 메모리
        MOV   CODE_SEG,CS      ;CS의 복사
        MOV   AX,ES            ;PSP 세그먼트 값의 복사
        MOV   BX,ZSEG          ;프로그램세그먼트 어드레스의 끝
        SUB   BX,AX            ;차이를 계산
        MOV   AH,4AH           ;SETBLOCK 기능번호
        INT   21H              ;실행
        JC    SETBLK_ERROR     ;캐리 플래그
;---메모리 할당의 오버레이
        MOV   BX,1000H               ;오버레이를 위해 1000H바이트 할당
        MOV   AH,48H                 ;메모리 할당의 기능 번호
        INT   21H                    ;할당 시도
        JC    ALLOCATION_ERROR       ;캐리 플래그
        MOV   OVERLAY_SEG,AX         ;오버레이 블럭의 세그먼트 번지 기억
;---코드 세그먼트에서 오버레이 오프셋 계산
        MOV   AX,CODE_SEG            ;코드 세그먼트 값에서
        MOV   BX,OVERLAY_SEG         ;  오버레이 값을 뺀다.
        SUB   BX,AX                  ;이제 BX는 16바이트 단위값을 지님
        MOV   CL,4                   ;16을 곱하는 효과를 얻기 위해
        SHL   BX,CL                  ;좌측으로 4회 시프트
        MOV   OVERLAY_OFFSET,BX      ;오프셋 저장
;---첫번째 오버레이의 로드
        MOV   AX,SEGBLOCK            ;파라메터 블록의 위치
        MOV   ES,AX
        MOV   BX,OFFSETBLOCK
        MOV   AX,OVERLAY_SEG         ;파라메터 블럭의 첫번째 워드에
        MOV   [BX],AX                ;  오버레이의 세그먼트 번지 설정
        MOV   [BX]+2,AX              ;오버레이 세그먼트를 재할당 벡터로 이용
        LEA   DX,PATH                ;DS:DX가 화일 패스를 나타냄
        MOV   AH,4BH                 ;EXEC 기능 번호
        MOV   AL,3                   ;오버레이 코드
        INT   21H                    ;오버레이 로드
        JC    LOAD_ERROR             ;문제 발생시 에러 루틴으로 점프

;---프로그램 실행 개시
                    .
                    .
                    .
   CALL  DWORD PTR OVERLAY_OFFSET   ;오버레이 호출
```

```
                        .              ;오버레이는 far프로시듀어이기 때문에
                        .              ;DWORD PTR을 이용
                        .
;---오버레이의 관찰
  DSEG      SEGMENT                    ;데이터 세그먼트의 셋업
                        .              ;스택 세그먼트의 스킵
                        .
  DSEG      ENDS

  CSEG      SEGMENT          PARA PUBLIC 'CODE'
  OVERLAY   PROC   FAR
            ASSUME CS:CSEG,DS:DSEG
            PUSH   DS                  ;프로그램의 DS기억
            MOV    AX,DSEG             ;오버레이의 DS 설정
            MOV    DS,AX
                        .
                        .
                        .
            POP    DS                  ;끝나면 처음 DS를 저장
            RET
  OVERLAY   ENDP
  CSEG      ENDS
            END
```

로는 할당된 메모리 블록의 패러그래프 번호를 위해 AX에 옮겨진 번호를 사용하라.

재배치 인자는 오버레이내에 재배치 가능한 항목을 계산하여 오프셋을 제공한다.

오버레이가 로드되는 곳에 패러그래프 번호를 사용하면 이 번호가 셋업되어 기능을 호출하고 오버레이는 로드된다. 간단히 중복화일에 대한 패스(Path)를 바꿔서 반복하여 그 기능을 호출하면 그때마다 다른 오버레이가 로드된다.

리턴(Return)에서 만일 캐리 플래그(Carry Flag)가 세트되면 에러가 발생한 것이고, 이때 에러코드는 AX에 옮겨진다. 이 코드는 기능 번호가 틀릴 경우는 1로, 화일을 찾을 수 없을 때는 2로, 디스크에 문제가 있다면 5로, 메모리가 불충분하면 8이 된다.

오버레이가 메모리내에 있다면 파프로시듀어(Far Procedure)로 액세스된다.

2중 워드 포인터(Double-Word Pointer)는 이 호출을 수용하기 위해 데이터 세그먼트내에 셋업되어야 한다.

오버레이의 오프셋은 코드 세그먼트와 오버레이 세그먼트의 차를 계산하여 여기에 16(패러그래프에서 바이트로 값을 바꾼)을 곱함으로서 구할 수 있다. 2개의 변수 OVERLAY_OFFSET와 CODE_SEG는 차례로 놓여 포인터가 정확하게 셋업되도록 한다. 일단 로드된 오버레이는 CALL DWORD PRT OVERLAY_OFFSET에 의해 호출된다.

오버레이는 그 자신의 데이터와 스택세그먼트를 갖는 완전한 프로그램으로 단순한 프로그램으로 간주될 수 있다. 오버레이가 호출되면 그 자신의 데이터 세그먼트의 세그먼트 값을 DS에 놓는다.

EXE형에서 COM형으로 전환하는 방법

어셈블리 언어 프로그래머들은 보통 EXE형식에서 COM형식으로 그들의 프로그램을 전환하는 옵션을 갖고 있다. EXE 화일은 재배치에 대한 정보를 포함하는 헤더 필드(Header Field)를 갖고 있으며, DOS가 프로그램을 로드하는 동안 프로그램내에 확실한 어드레스를 재배치한다.

이와는 다르게 COM화일은 재배치를 필요로 하지 않는 방식으로 셋업된다. 그 화일들은 로드된 프로그램이 메모리에 존재하도록 하는 형태이다. 이러한 이유로 EXE화일은 디스크내에 COM에 상당하는 것보다 적어도 768바이트 이상을 차지한다. 또한, COM화일은 재배치를 하지 않기 때문에 좀더 신속하게 로드시킬 수 있다.

재배치는 세그먼트 레지스터내에 존재하는 어드레스를 세트시키는 프로세스(Process)이다. 예를 들어 프로그램은 다음과 같은 코드로 데이터 영역의 시작점을 알려준다.

```
MOV DX,OFFSET DATA_AREA
MOV AX,SEG DATA_AREA
MOV DS,AX
```

DX내의 오프셋은 세그먼트 레지스터 DS에 관계한다. DS 자체내에 존재하는 값은 무엇인가? 프로그램 코드는 절대 어드레스(Absolute address)를 요구하지만 페래그러프 번호 DATA_AREA는 메모리내에 프로그램이 로드되는지, DOS에 의해 변하는지 다른 프로그램이 메모리 맨끝에 존속하고 있는지의 여부에 따라 존재하게 된다.

DOS가 프로그램을 로드할 때 반드시 메모리내의 어디에선가 프로그램이 시작된다. 이러한 이유로 프로그램이 링크될 때 모든 수행은 프로그램의 시작점으로부터의 오프셋으로서 어떤 세그먼트 값으로 셋업된다. DOS가 재배치를 수행할 때 프로그램 배치의 시작

점 값은 세그먼트 값에 더해져서 세그먼트 레지스터에 의해 요구되는 절대 배치(Absolute Location)를 제공한다. 그림 3은 재배치 프로세스를 보여준다.

COM화일은 이러한 "세그먼트 고정"을 필요로 하지 않고 쓰여져 있기 때문에 재배치를 할 필요가 없다. 프로그램내의 모든 일은 스택뿐만 아니라 모든 데이터를 포함하는 코드 세그먼트의 시작으로부터의 오프셋으로 셋업된다. 이러한 이유로 프로그램 전체는 65535 바이트 길이를 초과할 수 없으며, 이 값은 어드레싱이 취급할 수 있는 최대 오프셋이다.

COM화일은 모든 세그먼트 레지스터를 프로그램 세그먼트 프레픽스의 하부로 나타내지만, EXE화일과 비교해 보면 DS 및 ES는 똑같이 초기화되어 있고, CS는 PSP를 따르는 최초의 바이트로 세트된다. COM형태로 프로그램을 셋업시키기 위해서는 다음과 같은 규칙을 따라야 한다.

① 프로시듀어로서 프로그램을 셋업시키지 말라. 대신에 코드의 시작에서는 가령 STAT 와 같은 문을 넣고, END STAT와 같은 문으로 프로그램을 끝낸다.

② ORG 100H를 코드의 시작점에 놓아라. 이것은 코드의 원천점을 세트한다.(즉, 지시 포인터를 세트한다) COM프로그램은 100H에서 시작되는데, 이 값은 PSP 다음의 최초 바이트이다. 왜냐하면 CS는 100H보다 밑에 있는 PSP의 시작점에 세트되기 때문이다. 다른 곳에서 코드를 시작하려면 100H에 JMP 지시를 놓는다.

③ ASSUME문은 코드 세그먼트의 값과 일치시키기 위해 DS, ES, SS에 세트시키는데 예를 들면 ASSUME CS : CSEG, DS : CSEG, ES : CSEG와 같다.

④ 프로그램의 데이터는 코드에 손상을 입히지 않는한 프로그램내의 어느 곳에 위치해도 된다. 지금까지 접해보지 못한 데이터 항목에 기준이 설정된다면, 매크로 어셈블러는 그 첫번째 패스에서 에러를 발생하기 때문에 데이터로 프로그램을 시작하는 것이 좋다. 데이터를 건너뛰라는 JMP지시로 프로그램을 시작한다.

⑤ MOV AX SEG NEW_DATA와 같은 세그먼트 픽스-업(fix up)은 사용할 수 없다.

⑥ 스택 세그먼트는 초기코드내에서 생략된다. 스택 포인터는 프로그램에 의해 사용되는 64K 어드레스 공간의 맨위로 초기화된다.(스택은 메모리내에서 밑으로 갈수록 점점 증가한다). 64K보다 작게 구성되는 COM 프로그램에서 SS 및 SP는 변할 수 있다. 프로그램을 링크시킬 때, 링커는 스택 세그먼트가 없는 상태를 나타내는 에러 메시지를 제공하는데 이것은 무시해도 상관없다.

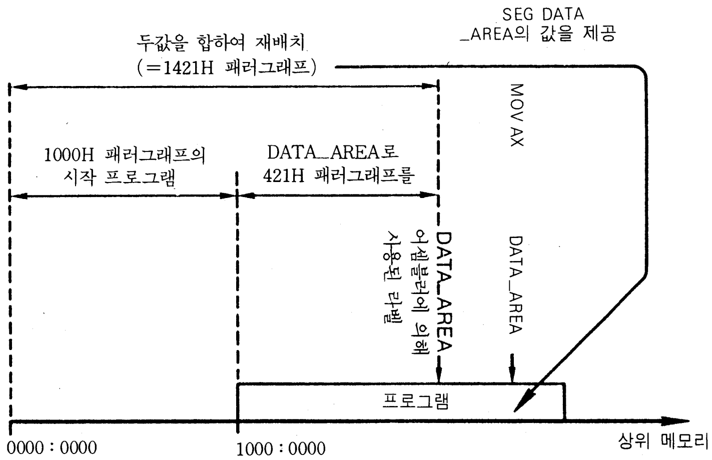
⑦ RET지시나 INT 20H를 써넣어 프로그램을 끝낸다. INT 20H는 프로그램을 끝낸 다음 제어를 DOS로 옮기는 기능이다. RET로 프로그램을 끝낼 때도 INT 20H를 사

〔리스트 7〕

```
 CSEG      SEGMENT
           ORG    100H
           ASSUME CS:CSEG, DS:CSEG,SS:CSEG
;---데이터:
 START:     JMP    SHORT BEGIN
 MESSAGE1   DB     'The dip switches are set for $'
 MESSAGE2   DB     'disk drive(s).$'
;---첫번째 메세지의 반을 프린트
 BEGIN:     MOV    AH,9                   ;기능 번호 9
            MOV    DX,OFFSET MESSAGE1     ;DX:DS에 스트링 번지 지정
            INT    21H                    ;스트링의 출력
            PUSH   AX                     ;다시 사용하기 위해 기능번호 유지
;---8255칩 A포트에  DIP스위치 세팅
            IN     AL,61H           ;B포트내의 바이트 설정
            OR     AL,10000000B     ;비트7을 ON
            OUT    61H,AL           ;바이트값으로 대치
            IN     AL,60H           ;A포트로부터 스위치를 세팅
            AND    AL,11000000B     ;상위 2비트를 분리
            MOV    CL,6
            SHR    AL,CL            ;ASCII기호로 변환시키기 위해 48을 더함
            ADD    AL,49
            MOV    DL,AL            ;DL에 AL값기억
            MOV    AL,61H           ;PB를 기억하고 그값를 선택
            AND    AL,01111111B     ;7비트 off
            OUT    61H,AL           ;바이트로 대치
;---드라이브의 번호 프린트
            MOV    AH,2             ;INT21H의 기능번호 2를 이용
            INT    21H              ;DL에 있는 번호 출력
;---두번째 메세지의 반을 출력
            POP    AX                     ;기능번호 선택
            MOV    DX,OFFSET MESSAGE2     ;다음 메세지를 출력할 준비
            INT    21H                    ;스트링 출력
            INT    20H                    ;프로그램의 종료
 CSEG       ENDS
            END    START

 C:\WP\GEMDISK3>
```

〔그림 3〕 DATA-AREA의 메모리위치의 재배치



용한다. 이것은 스택내의 최초 워드가 0로 초기화되었기 때문이다. 프로그램의 최종 RET지시에 접했을 때 0은 스택을 빠져 나와 지시 포인터를 프로그램 세그먼트 프레픽스의 시작점에 다시 연결시킨다. 그 배치에서 INT 20H 기능은 다음 지시로 실행된 다음 제어를 DOS로 돌린다. 이것 모두 PUSH DS / MOV AX, 0 / -PUSH AX과 같이 EXE화일에서 하는 것처럼 DS 및 0를 프로그램의 시작점에 넣을 수 없다.

프로그램을 이러한 방식으로 구성하여 어셈블 및 링크를 시켜야 한다. 그런 다음 DOS 디스켓에 들어 있는 EXE2BIN 을 사용하여 COM형식으로 전환시킨다. 만일 화일명이 링커에 의해 MYPROG.EXE로 주어진다면, 간단히 EXE2BIN MYPROG로 타이핑한다. 이것은 MYPROG.BIN으로 명명된 프로그램 화일을 생성한다. 이 시점에서 화일 MYPROG.COM로 이름을 바꾸는 것이 필요하다. 그렇지 않으면 .COM 확장된 화일로 전환시킬 때는 EXE2BIN MYPROG MYPROG.COM을 써 넣는다.

●어셈블리어의 경우

컴퓨터내에 얼마나 많은 드라이브가 있는지를 나타내는 딥스위치 세팅을 읽어 들이고 이것을 스크린에 나타내는 간단하면서 완전한 프로그램을 보여준다. 리스트 7은 COM형식이 이상적일 때 간단한 이용 프로그램의 분류에 대한 예이다. PC

SE I/O 포트

매킨토시 SE는 내부 FDD 2개, 외부 FDD 1개, RS-422 시리얼 포트 2개 및 내부 / 외부 SCSI 패러럴 포트를 접속시킬 수 있는 하드웨어 인터페이스를 갖추고 있다.

본회에서는 이들 인터페이스에 사용되는 하드웨어에 대해 설명하고, 내부, 외부 코넥터용 핀구성과 함께 IWM, SCC 및 SCSI 장치 들과의 소프트웨어 통신방법에 대해 알아본다. 또한 SCSI 데이터전송의 개념에 대해서도 간단히 살펴보겠다.

플로피디스크 인터페이스

SE에는 양면 800K FDD가 내장되어 있고, 추가로 같은 용량의 FDD를 하나 더 내장시킬 수 있게 되어 있다. 또한 외부 디스크드라이브 코넥터로써 양면 800K FDD와 편면 400K FDD 또는 애플 하드디스크 20을 접속시킬 수 있다.

애플 800K FDD는 800K 양면 및 400K 편면 플로피디스크에 포맷하면 리드, 라이트 할 수 있다.

FDD 포트는 IWM(Intergrated Woz Machine) 칩을 통해서 SE 메인로직 보드와 인터페이스가 가능하다. IWM은 디스크 스테이트 콘트롤 라인중 4개를 제어하며, 내부 / 외부 드라이브 선택신호 및 디스크 드라이브를 위한 데이터 리드 / 라이트신호를 발생시켜 준다. VIA는 디스크 스테이트 콘트롤 라인으로써, 디스크 드라이브 내부 레지스터 선택을 용이하게 하며 상위 내부디스크 드라이브와 하위 내부디스크 드라이브 사이를 선택하게 해준다.

SE 플로피디스크 인터페이스는 고전 매킨토시와 매킨토시 Ⅱ에서 사용되는 인터페이스와 거의 동일하고, Apple Ⅱ 및 Apple Ⅲ 컴퓨터에 쓰이는 인터페이스와도 유사하다.

표 1에 20핀 내부 플로피디스크의 코넥터핀에 대한 신호지정을 나타냈다.

IWM의 PH0, PH1, PH2 신호와 VIA의 SEL신호가 FDD의 레지스터를 선택하는데 이용된다. IWM의 PH3 신호는 FDD의 레지스터에 데이터를 라이트하는데 이용된다. 이들 신호에 대한 상세한 내용은 뒤에서 다룰「디스크 스테이트 콘트롤 라인의 제어」절을 참조하기 바란다.

/WRREQ 신호는 IWM의 프로그램 출력이 가능한 핸드세이크용으로 사용될 수 있는 것이다. /ENBL.HI와 /ENBL.LO 신호는 드라이브 선택신호이다. IWM은 /ENBL. EXT와 /ENBL. INT 신호를 발생시키며, 이에 의해서 각각 외부 드라이브 또는 내부 드라이브가 선택된다. /ENBL. INT 신호는 GLU 칩으로 송신되는데, 이 GLU 칩은 또한 VIA로부터 /HI. DRIVE 신호를 수신한다. /ENBL.INT

〔표 1〕 매킨토시 SE 내부 플로피디스크 코넥터 신호지정

핀 번 호	신 호 명	신 호 내 용
1	GND	Ground
2	PH0	Register select line CA0
3	GND	Ground
4	PH1	Register select line CA1
5	GND	Ground
6	PH2	Register select line CA2
7	GND	Ground
8	PH3	Register Write Strobe LSTRB
9	n. c.	Not connected
10	/WRREQ	Write Data Request
11	+5V	+5volts
12	SEL	Register select line SEL
13	+12V	+12volts
14	/ENBL. HI /ENBL. LO	Upper Internal Drive Enable ; or Lower Internal Drive Enable
15	+12V	+12volts
16	RD	Read Data
17	+12V	+12volts
18	WR	Write Data
19	+12V	+12volts
20	n. c.	Not connected

와 /HI.DRIVE 신호가 동시에 수신되면, GLU는 상위 내부드라이브에 /ENBL.HI 신호를 보낸다. 만약 /ENBL.INT 신호만 수신되고 /HI.DRIVE 신호는 수신되지 않을 경우에는 GLU는 하위 내부드라이브에 /ENBL.LO 신호를 보낸다.

외부 디스크 코넥터는 외부 FDD나 애플 하드디스크 20용으로 사용될 수 있다. 하드디스크 20과 외부 편면이나 양면 FDD를 동시에 사용하려면, 하드디스크20을 매킨토시 SE의 뒷면에 있는 코넥터에 꽂고, 외부 FDD를 하드디스크20 뒷면에 있는 DB-19 코넥터에 꽂으면 된다. 하드디스크20의 DB-19 코넥터 핀구성은 매킨토시 SE 뒷면에 있는 DB-19 디스크 코넥터 핀구성과 동일하다.

표 2는 매킨토시 SE 외부 플로피디스크 코넥터의 신호지정을 나타낸 것이다.

외부 플로피디스크 코넥터에 유용한 신호는, 표 1에서 설명했던 내부 플로피디스크 코넥터의 유용한 신호와 유사하다.

IWM은 디스크 드라이브와 통신을 위해 사용되는 NRZI 시리얼 데이터, 그리고 MC68000과 통신을 위해 사용되는 8비트 패러럴 데이터 사이에서의 변환을 해준다.

매킨토시 SE의 IWM에 의해 사용되는 클럭은 C16M신호(15.6672MHz)이다. 이 주파수는 기존의 매킨토시 IWM 클럭에서 사용되던 것에 비해 2배가 되는 주파수이다. 부트 시퀀스 도중에 애플 400K와 800K FDD 또는 하드디스크20에 액세스할 때는 IWM에서 1비트가 소프트웨어에 의해서 세트되어 15.6672MHz의 고속 클럭을 둘로 나누게 된다.

IWM은 데이터를 대략 최대 500K 비트/s로 디스크드라이브에 보낼 수 있다. GCR 인코드 데이터 4바이트는 RAM 시스템의 데이터 3바이트에 해당되기 때문에 최대 실용 데이터 전송률(Transmission Rate)은 대략 매초당 375K비트가 된다. 그러나 여기서 주목할 것은 디스크로의 데이터 처리율은 실제 데이터전송에 소요된 시간과, RAM을 리드하고 다른 시스템 명령을 수행하면서 데이터의 정확한 위치를 찾아내는데 걸리는 시간에 달려있다는 것이다.

〔표 2〕 매킨토시 SE 외부 플로피디스크 코넥터 신호지정

핀 번 호	신 호 명	신 호 내 용
1	GND	Ground
2	GND	Ground
3	GND	Ground
4	GND	Ground
5	−12V	−12volts
6	+5V	+5volts
7	+12V	+12volts
8	+12V	+12volts
9	n. c.	Not connected
10	PWM	Motor speed control
11	PH0	Register select line CA0
12	PH1	Register select line CA1
13	PH2	Register select line CA2
14	PH3	Register Write Strobe LSTRB
15	/WRREQ	Write Data Request
16	SEL	Register select line SEL
17	/ENBL.EXT	External Drive Enable
18	RD	Read Data
19	WR	Write Data

그림 1은 매킨토시 SE의 플로피디스크 인터페이스 회로도이다. 이 회로도에서 독자들은 매킨토시 SE 내부 플로피디스크 코넥터와 DB-19인 외부 디스크 코넥터의 핀구성을 볼 수 있을 것이다.

●디스크 스테이트 콘트롤 라인의 제어

IWM은 디스크 드라이브에 연결된 스테이트 콘트롤 라인을 소프트웨어적으로 제어할 수 있도록 하는 레지스터를 갖고 있다. 즉 매킨토시 SE 어드레스 공간에 특정장소를 지정(어드레싱)해 줌으로써 이러한 스테이트 콘트롤 라인을 ON, OFF시킬 수 있다.

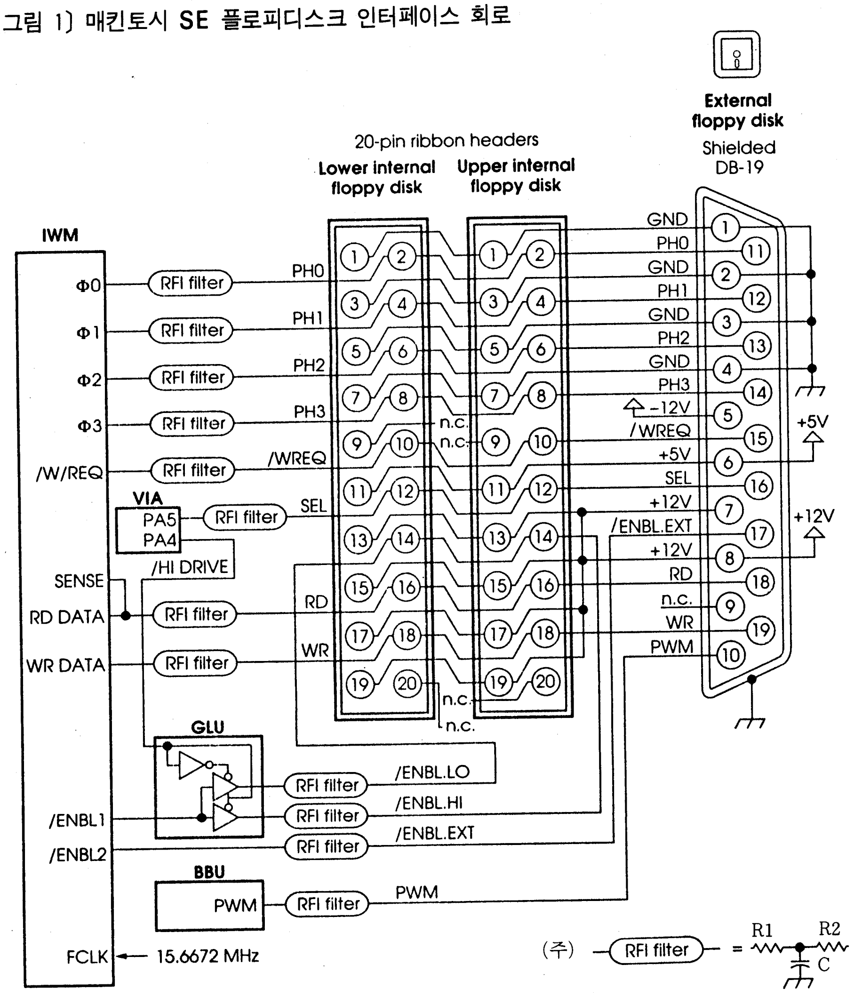
이 라인중 한 라인을 ON 또는 OFF 시키고자 하면 해당 장소에 어떠한 형태이든지 바이트 크기의 메모리를 액세스(리드 또는 라이트 기능)해주면 된다. 또 하나의 FDD 스테이트 콘트롤 라인인 SEL 라인은 VIA 데이터 레지스터 A의 vHeadSel 비트(비트5)에 의해 제어된다. /H1.DRIVE 디스크 드라이브 선택라인은 VIA 데이터 레지스터 A의 vDriveSel 비트(비트4)에 의해서 제어되며 2개의 내부 FDD가 있을 경우에는 VIA 데이터 레지스터 A의 비트 4의 값에 의해 하위 내부 FDD(vDriveSel=1)와 상위 내부 FDD(vDriveSel=0)가 선택된다.

CA0, CA1 및 CA2라인은 VIA로부터의 SEL라인과 함께 FDD내의 레지스터 신호와 데이터신호를 선택해 주는데 이용된다.

LSTRB라인은 디스크 레지스터에 제어정보를 라이트 해줄때 사용된다. SELECT는 어느 FDD가 인에이블 가능한가를 선택해 주는 IWM의 내부 소프트웨어이다.

셀렉트가 ON일 때는 외부 FDD를 선택해 주고 셀렉트가 OFF일 때는 내부 FDD를 선택해준다. 2개의 내부 FDD를 가질 경우에는 하위 내부 FDD와

〔그림 1〕 매킨토시 SE 플로피디스크 인터페이스 회로



〔표 3〕 매킨토시 SE IWM포트 콘트롤에서 dBase로부터 오프셋

IWM 라인	ON시켰을 때	OFF시켰을 때
Floppy disk drive state-control lines		
CA0	ph0H	ph0L
CA1	ph1H	ph1L
CA2	ph2H	ph2L
LSTRB	ph3H	ph3L
Floppy disk drive enable line		
ENABLE motor On	motor Off	
IWM Internal states		
SELECT	extDrive	intDrive
Q6	q6H	q6L
Q7	q7H	q7L

상위 내부 FDD 사이에서 선택해 줄 수 있도록 VIA 데이터 레지스터 A의 vDriveSel비트가 먼저 세트되어야 된다(표 3 참조).

인에이블라인은 선택된 FDD를 인에이블 시켜준다. IWM 내부 소프트웨어 스위치 Q6과 Q7은 디스크 레지스터의 정보를 리드하고, 또한 플로피디스크에 저장된 데이터를 리드/라이트 할 수 있도록 IWM의 내부 상태를 셋업시키는데 사용된다.

●디스크 레지스터로부터 리드하는 방법

디스크 드라이브의 여러 레지스터로부터 정보를 리드할 수 있는데 이로써 플로피디스크가 록되어 있는지의 여부와 플로피디스크가 드라이브내에 있는지의 여부, 헤드가 트랙 0번에 위치하고 있는지의 여부, 디스크 드라이브는 몇개의 헤드를 가지고 있는지의 여부 및 접속된 디스크 드라이브의 존재유무 등을 확인할 수 있다. 또한 이 레지스터중 일부는 헤드의 작동, 모터의 ON, OFF 및 플로피디스크의 이젝트(eject)를 수행하도록 라이트도 할 수 있다. 또 디스크 레지스터로부터 리드하기 전에는 데이터를 프로세서 데이터버스에 통과시킬 수 있도록 IWM 상태를 세트시켜야 된다.

디스크 레지스터를 액세스하기 위해 IWM을 셋업시킨 경우에는 리드하기를 원하는 레지스터를 선택해 주어야 하는데, 하나의 디스크 드라이브를 리드하고자 할 때에는 사용코자 하는 FDD를 인에이블 시킨 후에 LSTRB를 로우상태가 되게 한다. 그리고 CA0, CA1, CA2 및 SEL을 세트시켜 원하는 레지스터를 지정해준다.

단 디스크 레지스터로부터 데이터의 리드를 완료한 후에는 IWM의 상태는 FDD가 인지할 수 있는 상태로 만들어야 된다. 그렇지 않을 경우에는 다시 디스크 액세스를 시도할 경우에 시스템이 파괴될 수 있다.

IWM이 이용 가능한 로직 스테이트에 있게 하도록 디스크 레지스터의 리드가 끝난 후에는 항상 Q6을 다시 OFF시켜야 된다.

표 4는 여러 디스크 레지스터와 데이터 신호를 리드하기 위해서 디스크 스테이트 콘트롤 라인을 세트시킨 것이다.

〔표 4〕 리드하기 위한 매킨토시 SE IWM 디스크 스테이트 콘트롤 라인 세팅

스테이트 콘트롤 라인				세트된 레지스터	레지스터 내용
CA2	CA1	CA0	SEL		
0	0	0	0	DIRTN	Head step direction
0	0	0	1	CSTIN	Floppy disk in place
0	0	1	0	STEP	Floppy disk drive head stepping
0	0	1	1	WRTPRT	Floppy disk locked
0	1	0	0	MOTORON	Floppy disk drive motor running
0	1	0	1	TKO	Head at track 0
0	1	1	1	TACH	Tachometer
1	0	0	0	RDDATA0	Read data, lower head
1	0	0	1	RDDATA1	Read data, upper head
1	1	0	0	SIDES	Single-or double-sided drive
1	1	1	1	DRVIN	Floppy disk drive attached

●디스크 레지스터에 라이트하는 방법

디스크 레지스터에 라이트하기 위해서는 먼저 LSTRB를 0이 되게 하고 CA0와 CA1을 1로 세트시킨 후 SEL을 0으로 세트시킨다.

다음에 표 5에서 적절한 CA0와 CA1 값을 택하여 세트시키고 난뒤, CA2를 라이트하고자 하는 디스크 레지스터 값으로 세트시킨다.

LSTRB를 하이(1) 상태에서 최소 1㎲ 동안 유지시키는데, 플로피디스크를 이젝트시키지 않을 경우에는 최대 1ms 미만이 되게 한다. 그리고 다시 LSTRB를 로우상태로 한다. LSTRB가 하이상태일 때에는 CA0, CA1, CA2 또는 SEL을 변경시키면 안되고 SEL을 변경시키기 전에 CA1은 하이상태로 세트시켜야 된다는 것을 주의해야 한다.

표 4에서 보는 바와같이 우선 DIRTN 신호는 연속된 헤드의 스텝핑(stepping) 방향을 세트시켜 준다. 즉 0은 내부트랙(트랙번호 79)으로 향하도록 하고, 1은 외부트랙(트랙번호 0)으로 향하도록 해준다.

CSTIN은 디스크가 FDD에 존재할 때에만 0이 된다. STEP을 0으로 세트시키면 헤드가 DIRTN에 의해 가장 최근에 지정된 방향으로 모든 트랙을 한번 검색하게 해준다. 이 스텝이 완료되면(약 12ms) STEP은 다시 1로 세트되고 다시 스텝핑을 할 수 있게 된다.

WRTRRT는 플로피디스크가 라이트 보호상태일 때에는 항상 0상태이다. WRTRRT가 1상태가 아닐 때에는 FDD에 라이트하지 말아야 한다.

MOTORON은 FDD 모터의 상태를 제어해준다. 즉 0은 모터를 ON시키고 1은 모터를 OFF시킨다. 모터는 드라이브가 인에이블상태이고, 플로피디스크가 제자리에 있을 때에만 동작가능하다. 그렇지 않을 경우에는 라이트가 불가능하다. TKD는 헤드가 트랙번호 0에 위치할 때만 0상태가 된다. 이 동작은 헤드를 트랙번호 0에 위치시킨 후 처음 12ms 동안만 유효하다.

EJECT에 1을 라이트하면 플로피디스크를 드라이브로부터 이젝트시킨다. 플로피디스크를 이젝트시키기 위해서는 LSTRB를 최소 0.5초 동안 하이상태로 유지시켜야 된다.

동작시 FDD 속도는 TACH의 펄스트레인으로써 알아볼 수 있다. TACH 라인은 드라이브 모터 1 회전당 60펄스를 발생한다. 800K 양면 FDD는 자동적으로 액세스되는 트랙에 알맞는 속도로 조정된다.

400K 편면 FDD의 모터속도는 BBU에 의해 제어되는데 이 BBU는 PWM 포맷으로 RAM의 디스크 속도 버퍼로부터 인코드되는 정보(데이터)를 리드한다.

RDDATA0와 RDDATA1는 디스크 드라이브 헤드로부터의 순간적인 데이터를 전송한다.

SIDES는 항상 편면 FDD에서는 0, 양면 FDD에서는 1이다.

DRVIN은 선택된 FDD가 매킨토시 SE에 물리적으로 접속되었을 때에는 0, 그렇지 않을 때에는 1이다.

SE 시리얼 인터페이스

매킨토시 SE 시리얼 인터페이스(SCC)는 자일로그 Z8530 시리얼 커뮤니케이션 콘트롤러(Serial Communications Controller : SCC) 칩에 의해 제어된다. SCC 칩은 2개의 시리얼 포트를 보유한다. 즉 SCC 포트 A인 모뎀포트와 SCC포트 B인 프린터포트이다. 각각의 SCC포트는 뒷편 패널에 위치한 코넥터를 가지는데 이는 시리얼 페리패럴 장치의 접속과 컴퓨터를 AppleTalk 네트워크에 접속하기 위한 것이다.

●SCC 코넥터

매킨토시 SE는 컴퓨터 뒷편 패널에 있는 2개의 소형 8핀 코넥터를 이용하여 시리얼 페리패럴장치를 부착시킨다.

〔표 5〕 라이트하기 위한 매킨토시 SE IWM 디스크 스테이트 콘트롤 라인 세팅

스테이트 콘트롤 라인			세트된 레지스터	레지스터기능
CA1	CA0	SEL		
0	0	0	DIRTN	Set stepping direction
0	1	0	STEP	Step floppy disk drive head one track
1	0	0	MOTORON	Turn on/off floppy disk drive motor
1	1	0	EJECT	Eject the floppy disk

그림 2는 매킨토시 SE시리얼 코넥터의 핀구성이다.

표 6에 시리얼 코넥터의 신호지정을 나타냈다. 이 신호들에 대해서는 SCC 포트에서 설명하겠다.

● SCC 포트

매킨토시 SE 시리얼 인터페이스의 전송 및 수신데이터 라인은 일반적으로 많이 사용되고 있는 RS-232C 기준과는 다른 EIA 기준인 RS-422에 준한다. 이들 RS-232C와 RS-422의 차이점을 살펴보면 RS-232C 송신기는 하나의 코먼 그라운드에 따라 신호를 변조하고 RS-422 송신기는 동신호의 반전된 신호에 대해서 신호를 변조시킨다는 것이다. 또한 RS-232C는 수신된 신호가 그라운드에 대해서 논리 1(1 상태)이 되기에 충분한 음의 상태인지의 여부를 감지하는 반면 RS-422 수신기는 단지 둘중 어느 것이 더 음의 상태인지를 감지한다. 그러므로 RS-422 신호는 RS-232C 신호와 비교하여 노이즈나 간섭(interference)에 대해 영향을 더욱 적게 받고 거리에 의한 성능저하가 비교적 적다.

SCC의 시리얼 입·출력은 차동(differential) 라인 드라이브(26LS30)와 수신기(26LS32)를 통해 외부 코넥터에 접속된다. 라인 드라이브는 전송중에는 하이 임피던스 모드상태로 유지하며 다른 AppleTalk 장치들이 이 라인을 통해서 전송을 할 수 있게 한다. 드라이버는 해당되는 포트의 RTS 출력을 약하게 함으로써 동작한다.

매킨토시 SE의 A포트는 B포트와는 달리 동기모뎀을 위한 보조장치를 가지고 있다. SCC칩의 A포트 (즉, 모뎀포트)는 고속 우선순위방식을 채택하고 있기 때문에 고속도의 통신에 더욱 적절하다. A 포트의 고속 우선순위방식은 SCC의 내부처리 작동에만 영향을 미칠뿐 매킨토시 SE장치의 인터럽트 기법(scheme)에서 SCC에 대한 인터럽트 우선선택에는 영향을 미치지 않는다.

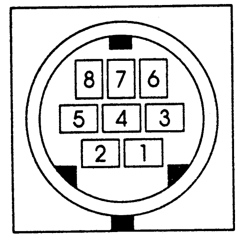
이 두가지 차이점을 제외하고는 A 포트와 B포트는 동일하다. 각각의 매킨토시 SE 시리얼포트의 출력 핸드세이크 신호(HSKo)는 해당 포트인 SCC의 DTR(Data Terminal Ready) 출력에서 비롯된다.

각 포트의 입력 핸드세이크신호(HSKi) 는 해당포트인 SCC의 CTS(Clear To Send) 입력과 접속되어 있고 내부장치의 DTR 핸드세이크 신호를 받아들이도록 설계되어져 있다. 이 라인은 또한 해당포트인 SCC의 TR$_x$C(Transmit / Receive Clock) 입력과도 접속되어 있어 외부장치가 고속도로 동시에 데이터를 교환할 수 있도록 해준다.

주의할 것은 HSKi라인을 고속의 데이터클럭을 수신하는데 사용하는 중이라면 DTR을 수신하는데는 이 라인을 사용할 수 없다. 또한 각 포트는 범용입력(GPi, 7번핀)을 갖추고 있는데 이는 해당포트인 SCC의 DCD(Data Carrier Detect)에 접속되어 있다. 이 입력은 외부장치에서 매킨토시 SE로 핸드세이크 신호를 주는데 이용될 수 있다. SCC에 대한 DCD입력은 소프트웨어에 의해서 폴링(polling)이 이루어질 수도 있고 CPU 인터럽트를 발생시키는데 이용될 수도 있다.

SCC는 3.672MHz 클럭을 SCC의 PCLK핀과 각 채널의 Receive / Transmit 클럭입력을 통해 수신한다. 3.672MHz 클럭은 SCC의 내부 보오 레이트(baud rate) 콘트롤러에 의해 사용된다. 매킨토시 SE는 그라운드된 음의 입력과 함께 표준 RS-422 수신기(26LS32 칩)중 하나의 양의 (비반전된) 입력을 이용하여 입력 핸드세이크 신호(HSKi)를 감지해 낸다.

〔그림 2〕 매킨토시 SE 시리얼코넥터의 핀구성



범용 입력(GPi)은 양의 입력을 접지시키고 똑같이 차동 수신기중 하나의 음(반전)의 입력을 이용하여 수신된다.

RS-422 수신기(26LS32)는 차동수신기이기 때문에 이를 구동시키는 모든 핸드세이크나 클럭신호는 내부적으로 접지된 입력에 대해서 음의 전압과 양의 전압사이에서 변화되는 형태이어야 된다. 만약 어떠한 장치나 핸드세이크 로직중 하나로써 그라운드(0V)를 사용하려고 시도한다면 매킨토시 SE는 이를 미결정 상태로 판단하고 결과에 대해서는 예상할 수 없게 된다.

SCC칩은 시리얼 라인을 통해 입·출력하는 동안 레벨 2의 프로세서 인터럽트(/ IPL 1)를 발생한다. 디스크 액세스동안에는 CPU가 인터럽트에

〔표 6〕. 매킨토시 SE 시리얼코넥터 신호지정

핀 번 호	신 호 명	신 호 내 용
1	HSKo	Output Handshake
2	HSKi	Input Handshake or external clock
3	TxD−	Transmit Data−
4	GND	Signal ground
5	RxD−	Receive Data−
6	TxD+	Transmit Data+
7	GPi	General-purpose input
8	RxD+	Receive Data+

대한 작동을 멈추고 있을 때 발생할 수 있는 데이터 손실을 방지하기 위해서 레벨 3 이하의 모든 인터럽트를 디스크 콘트롤러가 디스에이블 시켜준다.

그렇지만 SCC 칩은 3바이트가 넘는 수신데이터는 저장시킬 수 없다. 그러나 AppleTalk 메시지와 같은 시리얼 데이터는 항상 들어올 수 있기 때문에 플로피디스크 작동이 완료되기 전에도 계속해서 들어오는 데이터를 받아들일 수 있도록 SCC는 계속 작동할 필요가 있다.

그러므로 소프트웨어로 하여금 리드를 하기 위해 기다리고 있는 시리얼데이터 바이트의 유무여부를 결정할 수 있도록 VIA는 SCC의 웨이트/리퀘스트라인을 감시해준다. 플로피 디스크동작이 멈출 때마다 디스크 드라이버는 VIA 데이터 레지스터A의 7번 비트를 체크하여 웨이트/리퀘스트라인의 상태를 결정해줄 수 있다.

만약 SCC에서 대기상태의 데이터가 있을 경우에는 드라이버가 이 대기 데이터를 다음 디스크 작동이 멈출 동안에 리드한다. 매킨토시 Toolbox중 최대의 데이터 전송률은 57,600 보오(baud) 이다. 이 전송률이 SCC 포트가 3.672MHz 클럭 시간에서 비동기식 인터럽트 구동양식으로 작동될 때에 매킨토시 SE컴퓨터가 시리얼데이터 전송에서 얻을 수 있는 최대 전송률이다.

매킨토시 SE컴퓨터에서 얻을 수 있는 최대 시리얼데이터 전송률은 약500K 보오이다. 이 데이터 전송률을 얻기 위해서 SCC는 외부 클럭에 의해 시간이 조정되는 동기모드에서 작동되어야만 하고 시리얼 드라이버는 컴퓨터에 의해 계속적으로 완전하게 제어되어야 한다.

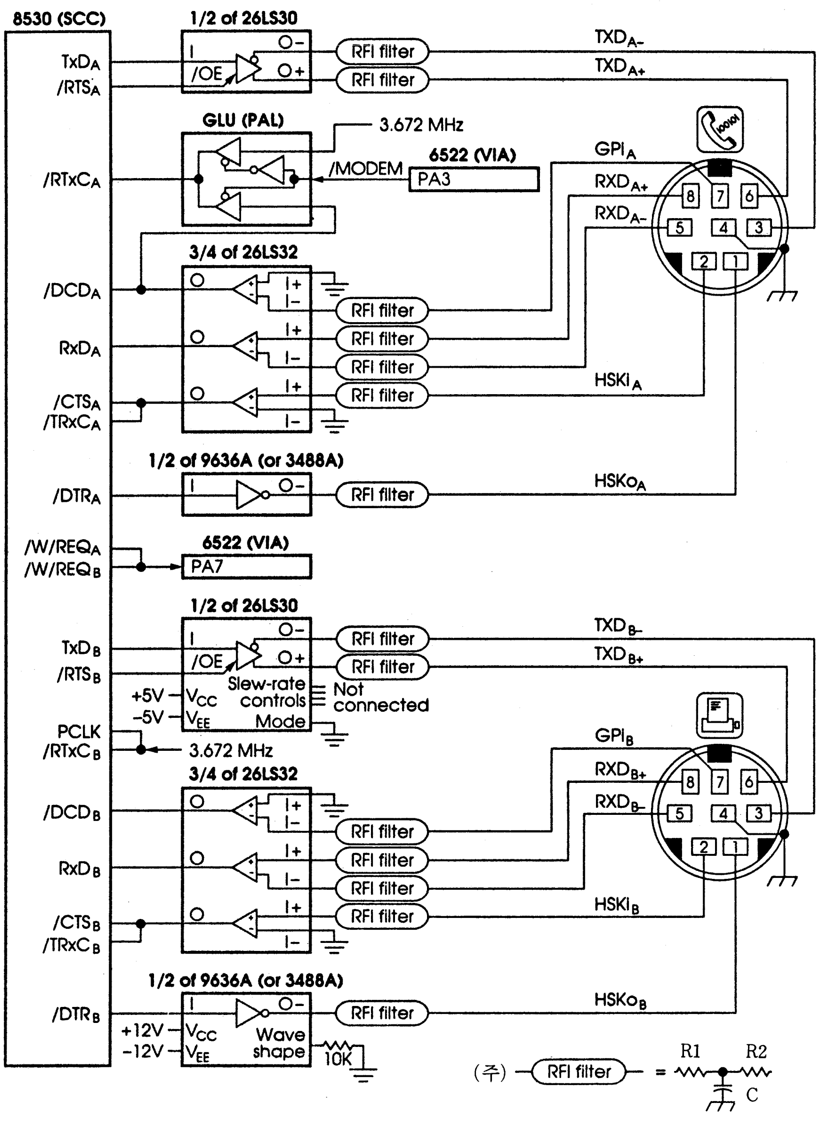
SCC 콘트롤 및 데이터 레지스터 위치를 표 7에 나타냈다. 이 표에서 주어진 위치 오프셋을 사용하면 각각의 액세스를 위한 올바른 짝수 및 홀수 어드레스가 자동적으로 제공된다.

SCC로부터 데이터를 리드하고자 할

〔표 7〕 SCC 레지스터의 위치

위 치	레 지 스 터
sccWBase+aData	Write Data register A
sccRBase+aData	Read Data register A
sccWBase+bData	Write Data register B
sccRBase+bData	Read Data register B
sccWBase+aCtl	Write Control register A
sccRBase+aCtl	Read Control register A
sccWBase+bCtl	Write Control register B
sccRBase+bCtl	Read Control register B

〔그림 3〕 매킨토시 SE 시리얼 인터페이스 회로



때는 짝수 어드레스을 액세스해야만 하고 SCC에 데이터를 라이트하고자 할때에는 홀수 어드레스을 액세스해야 만된다. SCC는 오직 데이터버스의 상위 바이트에만 위치하나 이러한 액세스 동작이 이루어지는 이유는 MC68000이 짝수 어드레스로부터 리드할 때에는 데이터버스의 상·하위 바이트 모두에 같은 데이터를 라이트할 뿐 아니라 홀수 어드레스로부터 리드함으로써 SCC를 리세트 시켜주기 때문이다.

그림 3은 매킨토시 SE 시리얼 인터페이스의 회로도이다.

소형 컴퓨터시스템 인터페이스

매킨토시 SE는 NCR 5380 소형 컴퓨터시스템 인터페이스(SCSI) 칩에 의해 제어되는 고속 페러럴 통신포트를 갖고 있다. NCR 5380은 애플 하드디스크인 SC와 기타 SCSI HDD, 스트리밍(Streaming) 테이프 드라이브 및 고속프린터와 같은 SCSI의 주변장치를 7개까지 통신을 할 수 있는 용량을 갖추고 있다. 매킨토시 SE는SCSI 주변장치를 부착할 수 있도록 병렬로 접속된 2개의 코넥터(즉 내부 50핀 플랫 리본 코넥터와 외부 DB-25 코넥터)를 갖추고 있다.

만일 SCSI 네트워크가 어떠한 주변기기를 접속시켜 사용된다면 최소한 한세트의 단말저항기가 부착되어야 한다.

SCSI 네트워크는 오픈콜렉터 로직을 사용하고 있으며 단말 저항기는 약 3V의 불활성레벨에서 SCSI 버스를 정확하게 업 시키는데 필요하다.

만일 내부 또는 외부장치에 단말저항기가 없을 경우에는 최소한 하나의 애플 터미널 블록을 SCSI 네트워크에서 적절한 위치에 설치시켜야 된다. 즉 첫번째는 SCSI 주변장치가 매킨토시 SE에접속되면 단말저항기도 역시 부착되어야 되고 (주변기기 내부나 외부에 애플터미널 블록의 형태로써), 또 이 저항기를 위한 전력이 주변기기나 어떠한 외부전원에 의해 제공되어야만 된다는 것을 의미한다.

또한 전력공급의 제한 때문에 매킨토시 Plus컴퓨터는 SCSI코넥터의 단말저항기를 위한 전력을 공급시켜 주어야만 된다.

매킨토시 SE SCSI 포트는 ANSI X3T9.2 표준과 다음 두가지 면에서 다르다.

① 매킨토시 SE는 외부 SCSI 주변기기를 부착시키는데 DB-25 코텍터를 사용한다. ANSI X3T9.2 표준 (50핀 플랫 리본 코넥터)은 단지 내부 SCSI 주변기기를 위해서만 매킨토시가 SE에 의해 이용된다. 그러나 외부 코넥터가 오직 25핀만을 사용하더라도 ANSI X3T9.2에서 규정하는 모든 신호와 콘트롤라인을 얻을 수 있다. 단, 엑스트라 그라운드 라인만이 제외되어 있다. DB-25 외부 코넥터에서의 신호는 내부 50핀 플랫 리본 코넥터에 의해 사용되는 신호와 동일하다. 내부 및 외부 코넥터는 병렬로 연결되어 있으며 둘다 모두 NCR5380 칩에 의해 제어된다. 즉 2개의 코넥터사이에 데이터 전송률이나 성능에서의 차이점은 없다. SCSI 주변기기에 표준 50핀 코넥터를 사용해야 한다면 애플어댑터 케이블을 이용하여 외부 DB-25 코넥터를 표준 50핀 코넥터로 변환시켜 주면 된다.

② 대부분의 경우에 내부 SCSI장치는 내장 단말저항기를 갖추고 있다. 매킨토시 SE는 두가지 SCSI코넥터 모두에 단말저항기를 위한 전력을 공급해준다. 그러나 단말저항기를 제공해 주지는 않는다.

●SCSI 코넥터

표 8은 SCSI DB-25 외부 코넥터의

〔표 8〕 매킨토시 SE SCSI 외부 코넥터 신호지정

핀 번 호	신 호 명	신 호 내 용
1	/REQ	Request for a REQ/ACK data transfer handshake
2	/MSG	Indicates the message phase
3	/I/O	Controls the direction of data movement
4	/RST	SCSI bus reset
5	/ACK	Acknowledge for a REQ/ACK data transfer handshake
6	/BSY	SCSI data bus is busy
7	GND	Ground
8	/DB0	Bit 0 of SCSI data bus
9	GND	Ground
10	/DB3	Bit 3 of SCSI data bus
11	/DB5	Bit 5 of SCSI data bus
12	/DB6	Bit 6 of SCSI data bus
13	/DB7	Bit 7 of SCSI data bus
14	GND	Ground
15	/C/D	Indicates whether control or data is on the SCSI bus
16	GND	Ground
17	/ATN	Indicates an attention condition
18	GND	Ground
19	/SEL	Select a target or an initiator
20	/DBP	Parity bit for SCSI data bus
21	/DB1	Bit 1 of SCSI data bus
22	/DB2	Bit 2 of SCSI data bus
23	/DB4	Bit 4 of SCSI data bus
24	GND	Ground
25	TPWR	+5 volts termination power

신호지정이다. 그리고 표 9는 SCSI 50핀 플랫 리본 내부 코넥터의 신호지정을 나타낸다.

RS-232C장치는 DB-25 SCSI 포트 코넥터에 접속시키면 안된다. 왜냐하면 SCSI인터페이스는 0V와 +5V의 표준 TTL로직을 사용하지만 RS-232C 장치는 일부 라인에서 −25V와 +25V의 로직을 사용하기 때문인데 이러한 전압은 매킨토시 SE 로직기판의 NCR5380 칩을 파괴시킨다.

●SCSI 포트

SCSI의 각 장치는 0에서 7까지의 고유장치로 ID(인식표)를 가진다. 즉 매킨토시 SE 컴퓨터의 ID는 항상 7이며 만약 장치 하나가 설치되었다면 내부SCSI 장치의 ID는 항상 0이다. SCSI 콘트롤 및 데이터 레지스터의 위치는 표 10과 같이 라이트에 대해서는 일정한 scsiWr로, 리드에 대해서는 scsiRd로부터 오프셋 된다.

SCSI로부터 데이터를 리드하기 위해서는 짝수 어드레스를 액세스해야 되고, SCSI에 데이터를 라이트할 경우에는 홀수 어드레스를 액세스하여야만 된다. SCSI는 오직 데이터버스의 상위 바이트에만 존재한다.

그림 4는 SCSI 인터페이스의 회로도이다.

●SCSI 데이터 전송

SCSI 데이터 전송은 CPU에 의해 제어되고 NCR5380 SCSI에 의해 동작된다. SCSI 칩은 두 모드에서 작동가능하다. 즉 노말 모드와 pseudo-DMA 모드 에서 작동이 가능하다.

노말 모드에서 SCSI 드라이버 소프트웨어는 주변 SCSI 장치와 통신을 하기 위해 필요한 모든 핸드세이킹을 제어해야 된다.

일반적인 SCSI 데이터 전송에서 CPU는 정보의 블록전송을 셋업시키기 위해 SCSI칩의 노말 모드를 사용한 후에 실제 데이터 전송을 실행시키는 데는 pseudo-DMA 모드를 사용한다. 데이터 전송이 일단 시작되면 SCSI칩은 주변장치로부터 1바이트를 수신할 때와 주변장치에 1바이트를 라이트시킬 때를 지시하기 위해서 SCSI칩의 버스와 스테이터스 레지스터의 DRQ비트를 이용한다.

매킨토시 SE는 주변기기의 허용속도보다 CPU가 더 빨리 데이터를 라이트하거나 타당하지 않는 데이터를 리드하는 것을 방지하기 위해서 pseudo-DMA모드의 하드웨어 핸드세이킹을 갖추고 있다. 이 핸드세이킹은 BBU칩에 의해 조정되며 이 칩은 SCSI칩의 DRQ라인이 하이상태가 될 때까지 각 바이트 전송을 완료하지 않는다. 이 핸드세이킹은 pseudo-DMA모드의 리드 및 라이트 동작을 매우 안전하게 해주

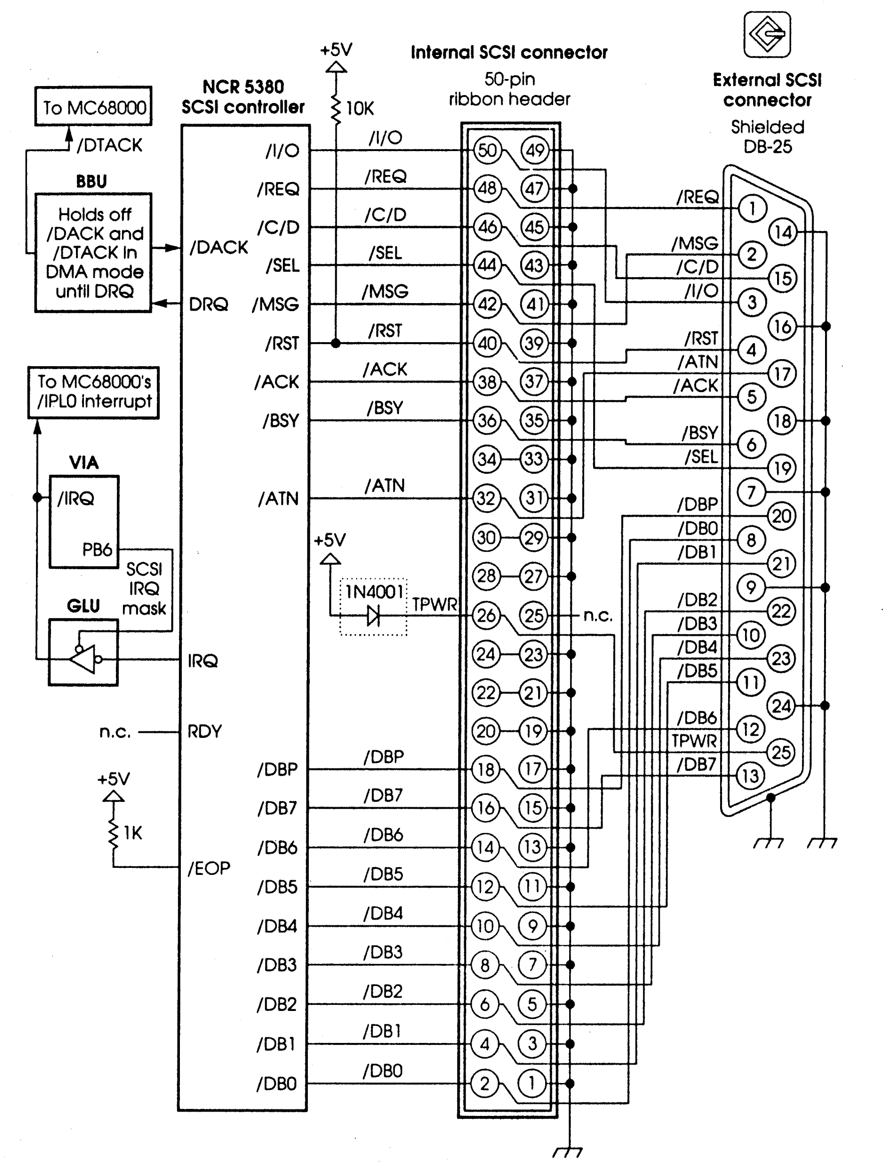
〔표 9〕 매킨토시 SE SCSI 내부 코넥터 신호지정

핀번호	신호명	의 미	핀번호	신호명	의 미
1	GND	Ground	26	TPWR	+5 volts termination power
2	/DB0	Bit 0 of SCSI data bus	27	GND	Ground
3	GND	Ground	28	GND	Ground
4	/DB1	Bit 1 of SCSI data bus	29	GND	Ground
5	GND	Ground	30	GND	Ground
6	/DB2	Bit 2 of SCSI data bus	31	GND	Ground
7	GND	Ground	32	/ATN	Indicates an attention condition
8	/DB3	Bit 3 of SCSI data bus	33	GND	Ground
9	GND	Ground	34	GND	Ground
10	/DB4	Bit 4 of SCSI data bus	35	GND	Ground
11	GND	Ground	36	/BSY	SCSI data bus is busy
12	/DB5	Bit 5 of SCSI data bus	37	GND	Ground
13	GND	Ground	38	/ACK	Acknowledge for a REQ/ACK data transfer handshake
14	/DB6	Bit 6 of SCSI data bus	39	GND	Ground
15	GND	Ground	40	/RST	SCSI bus reset
16	/DB7	Bit 7 of SCSI data bus	41	GND	Ground
17	GND	Ground	42	/MSG	Indicates the message phase
18	/DBP	Parity bit for SCSI data bus	43	GND	Ground
19	GND	Ground	44	/SEL	Select a target or an initiator
20	GND	Ground	45	GND	Ground
21	GND	Ground	46	/C/D	Indicates whether control or data is on the SCSI bus
22	GND	Ground	47	GND	Ground
23	GND	Ground	48	/REQ	Request for a REQ/ACK data transfer handshake
24	GND	Ground	49	GND	Ground
25	n. c.	Not connected	50	/I/O	Controls the direction of data movement

〔표 10〕 SCSI 레지스터 위치

위 치	레 지 스 터	위 치	레 지 스 터
scsiWr+sODR+dackWr	Write Output Data register in pseudo-DMA mode	scsiRd+sTCR	Read Target Command register
scsiRd+sIDR+dackRd	Read Input Data register in pseudo-DMA mode	scsiWr+sSER	Write Select Enable register
scsiWr+sODR	Write Output Data register in normal mode	scsiRd+sCDR	Read current SCSI data register
scsiRd+sIDR	Read Input Data register in normal mode	scsiRd+sCSR	Read current SCSI bus status
scsiWr+sICR	Write Initiator Command register	scsiRd+sBSR	Read Bus and Status register
scsiRd+sICR	Read Initiator Command register	scsiWr+sDMAtx	Start DMA send
scsiWr+sMR	Write Mode register	scsiWr+sTDMArx	Start DMA target receive
scsiRd+sMR	Read Mode register	scsiWr+sIDMArx	Start DMA initiator receive
scsiWr+sTCR	Write Target Command register	scsiRd+sRESET	Reset parity/interrupt

〔그림 4〕 매킨토시 SE SCSI 인터페이스 회로



며 심지어는 블라인드 전송모드를 사용할 때에도 안전하게 해준다.

만일 SCSI버스에서의 리드 또는 라이트 동작이 265ms이상동안 지연되면 SCSI칩은 DRQ에 신호를 주지 않으며 BBU칩이 MC68000에 버스 에러(/BERR)를 지정한다. 유사한 핸드세이킹이 매킨토시II에서 이용되지만 매킨토시 Plus에서는 이용되지 않는다.

매킨토시 SE의 한 블록내에서의 최대 SCSI 전송률은 폴링 전송에서는 약 172K/s이며 블라인드 전송에서는 약 65K/s이다.

그림 4에서와 같이 NCR5380 인터럽트 신호(IRQ)는 레벨 1 인터럽트 라인(/IPL0)에서 CPU를 인터럽트할 수 있다. VIA IRQ 마스크 vH4비트는 이 인터럽트를 디스에이블시키는데 이용될 수 있다.

□ 알림 □

지금까지 본 연재를 애독해주신 독자여러분께 감사드립니다. 본 매킨토시 아키텍처는 이번호를 마지막으로 끝을 맺게 되었음을 알려드립니다.

지성컴퓨터
G-TECH MICRO SYSTEM

代表 洪 斗 杓
서울시 종로구 장사동 175-4
세운상가 가동 4층 바열 419호
TEL : 277-9880

最高의 品質과 最適의 價格
그리고 最大의 滿足을 드립니다.

- 16BIT PC/XT/AT/386 APPLE Ⅱ. MSX 전기종 각종 주변기기 및 소모품
- HDD-20/30/40/80/120MB
- 행정전산망용 한글 한자 BOARD
- EGA, MULTI SYNC MONITOR 및 CARD

안녕하십니까!
그동안 저희 마이크로를 이용해주신
여러분께 감사드리오며, 보다 좋은 여건에서
손님 여러분을 맞이하고자 새로이 전자랜드에
판매장을 OPEN하게 되었음을 알려드립니다.
새해에도 여러분과 함께 Computer의
정상을 달리도록 노력하겠읍니다.

- IBM Compatible Computer
- 386 System, AT/XT System
- HARD Disk 전문
- Controller 전문
- Microsoft Mouse 지정대리점

마이크로컴퓨터시스템
용산 전자랜드 A-335 TEL : 719-4910~1
세운상가 가동 라열 423호 TEL : 274-4296, 265-3787

국민에 의한 국민을 위한 국민 컴퓨터

취급품목
IBM AT · XT
Apple Ⅱ
Msx 각종 주변기기

A/S 철저

국민컴퓨터
서울특별시 종로구 종로3가 175-4 (세운상가 가동 가열 405호)
TEL : (02) 265-0734

용산 전자랜드에
고도 컴퓨터 전시장 개장

- 업무용 및 교육용 컴퓨터
- Software (OA/FA)
- 금성 U.P.S (무정전전압조정장치) 전문점
- 금성 디스켓 총판장

고도컴퓨터시스템
용산매장 : 714-8181~2
종로매장 : 274-5337, 4311

FAITH COM

서울시 종로구 종로3가 175-4
세운상가 가동 라열 415호
● 판매부 : 277-6151 ● 공 장 : 272-1495

FAITH-AT
CPU : INTEL 80286(8/10/12/16MHz)
Co-Processor : (선택)
RAM : 1M(BASE 640KB, EXT 384KB)
ROM : 64KB, 256KB
Display : Mono-(선택 : COLOR)
보조기억장치 : FDD(1.2Mbyte×1) HDD(20Mbyte×1)
BIOS : Phoenix, Award
Ext-Slot : 8개
Key Board : 84, 101, 102Key
Power : 230W~250W
*XT, Apple compatible 취급

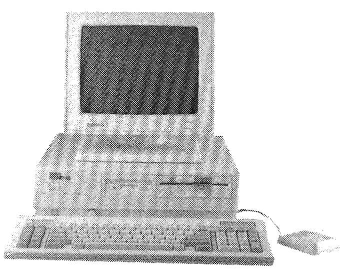
첨단기능의 16Bit 컴퓨터
TECHNOVA 저가격, 고성능

- **취급품목**
- TECHNOVA-XT/88+
- TECHNOVA-AT/286
- SOFTWARE R&D

테크노바컴퓨터
TECHNOVA COMPUTER
본 사 : 서울시 영등포구 여의도동 서린B/D 707-2
TEL : 783-9884
판매장 : 서울시 용산구 한강로3가(전자랜드RM 309-310)
TEL : 701-2443/2444

신제품

슬림형 업무용 컴퓨터
POWER 88



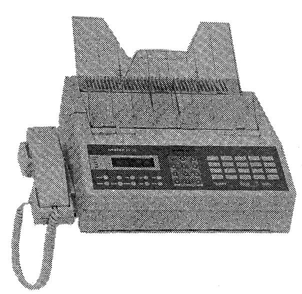
이 제품은 5.25인치와 3.5인치 FDD의 기본 내장으로 필요에 따라 드라이브의 선택사용이 가능하며 한글, 한자 MS-DOS와 OA S/W, 인터페이스 S/W, 각종 언어 등 풍부한 소프트웨어를 제공한다.

제품의 규격을 보면 다음과 같다.

- CPU : Intel 8088-1
- 속도 : 9.54 / 4.77MHz
- Co-Processor : 8087(선택)
- 주기억장치 : ROM : 16KB
  RAM : 640KB
- 한자 : 4,880자(완성형), 5,170자(조합형)
- 문자포트 : 영문, 숫자 : 8×16도트
  한글, 한자 : 16×16도트
- 내장 디스크 : 5.25인치(360KB×1) +3.5인치(1.44MB×1)
- 슬롯 : 4개
- 내장기구 : 시리얼포트×1
  패러렐포트×1
  클럭 / 캘린더 기능
  마우스 및 게임포트
- 모니터 : 12인치, 14인치(모노, 칼라)
- 그래픽 해상도 : 640×480, 640×400, 640×200, 320×200
- 키보드 : 86키, 101키

문의처 : 효성컴퓨터 사업부
전 화 : (02)545-4151

다양한 용도의 팩시밀리
DF-30



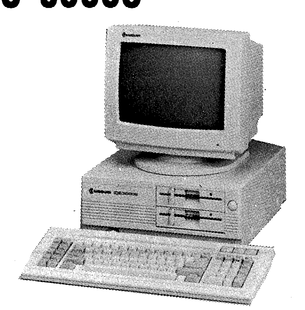
이 제품은 온후크다이얼, RS-232C 인터페이스, 전산용지송신, B_4수신, 16단계로 색을 분해하여 전송, 대화식메뉴, 50M 기록지내장, 자동컷 기능 등을 갖추고 있다.

제품의 규격은 다음과 같다.

- 사용회선 : 전용선, 일반가입회선
- 형식 : 송수신 겸용형
- 최대원고크기 : 전산용지(279㎜)
- 유효화면폭 : B_4(252㎜), A_4(206㎜)
- 전송속도 : 9600 / 7200 / 4800 / 2400bps
- 전송시간(A_4표준원고) : 10초
- 주주사선밀도 : 8Dot / ㎜, 4Dot / ㎜
- 부주사선밀도 : 7.7Line / ㎜, 3.85Line / ㎜
- 대역압축방식 : MH, HR, HS Line Skip
- 주사방식 : 평면주사
- 기록방식 : 감열기록
- 변조방식 : (고속모드) QAM, PHM
  (중속모드) AM-PM-VSB
- 동기방식 : (고속Mode) 전송동기
  (중속Mode) 독립동기
- 전원 : AC110V, 60Hz
- 소비전력 : 송신기 45W
  수신기 60W
  COPY시 75W
- 외형 : 371(W)×384(D)×130(H)㎜
- 중량 : 8㎏

문의처 : 대우통신
전 화 : (02)771-35(교환 : 4332~35)

저가의 보급형 컴퓨터
SPC-3000S



이 제품은 PC의 대중화 추세에 발맞추어 개발되었으며 교육용, 개인사무용, 단위 업무용, 정보 검색용, 단말기용으로 쓰인다.

관련 소프트웨어로는 교육, 취미용에 어셈블러, 베이직, 코볼, 포트란, C 및 각종 유티릴티, 개인 사무용에 MY-워드, OA-II 글벗 OA-II 샘벗, MY

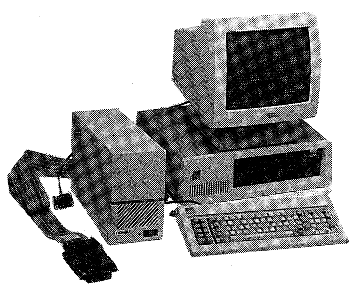
–시트, MY–이미지, MY–팩스, 단위 업무용에 금융, 무역, 운수, 각종 업무용 소프트웨어, 정보 검색용에 SECON(생활 정보 및 전자 사서함) 등이며, 단말기용으로는 IBM, PRIME, HP, UNIVAC, UNIX SYSTEM 등의 EMULATOR, LAN 등으로 지원가능하다.

주요 규격을 보면 다음과 같다.

- CPU : INTEL 8088–II(8MHz)
- RAM : 기본 256KB(확장 640KB)
- I/O 인터페이스 : 프린터용 패러럴 1포트
- 디스플레이 : 한글 CGA 조합형
- 키보드 : 84키

문의처 : 삼성전자 컴퓨터시스템 사업본부
전 화 : (02) 559–2643~6

8051 전용 에뮬레이터
EMUL51-PC



EMUL51-PC는 8051계열의 마이크로콘트롤러 H/W 및 S/W의 개발을 위한 고성능의 인서키트 에뮬레이터로 IBM PC/XT/AT 및 호환기종의 내부버스에 직접 내장 가능한 플러그인 보드로 되어 있으며, 다양한 트리거 능력과 함께 진보된 트레이스 기능은 물론 C, 파스칼과 같은 고급언어내에서의 개발을 위한 모든 기능을 제공하여 소스코드내에서 직접 에뮬레이션도 가능하게 하였다.

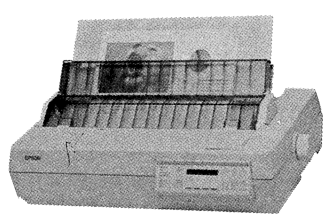
또한 풀-다운 메뉴, 마우스 사용, 초보자를 위한 온-라인 HELP기능을 갖추고 있으며, 표준 마이크로콘트롤러 칩외에 SECOND-SOURCE로 INTEL, OKI, SIEMENS, PHILIPS, SIGNETICS, AMD, DALLAS 등의 마이크로콘트롤러 칩도 지원하고 있다.

이 제품의 규격은 아래와 같다.

- 적용 시스템 : IBM PC/XT/AT, PS/2 또는 호환기종
- 실시간 에뮬레이션 메모리 : 64KB 코드 메모리/64KB 외부 데이터 메모리
- 브레이크 포인트 : 192,000EA(하드웨어)
- 프로그램 수행 분석 : 히스토그램, 통계적 정보
- 실행 타임 : 750μs이하
- 심볼릭 디버깅 기능
- 라인 어셈블러 및 디스어셈블러 기능
- 매크로 기능

문의처 : 제우스컴퓨터
전 화 : (02) 784–7841

초고속 24핀 칼라프린터
LQ-2550H



LQ–2550H는 흑백은 물론 여러가지 색상의 프린팅이 가능한 칼라프린터로서 7가지 칼라가 기본적으로 제공되고, 추가장치없이 원하는 색상을 선택할 수 있으며, 여러가지 폰트를 가지고 있어 한글 명조체, 고딕체 및 한자 명조체, 영문 Roman, Courier, Script, Prestige, Sans Serif 등 사용자가 원하는 글씨체를 선택하여 프린트할 수 있음은 물론 문서작성에 따른 출력양식도 다양하게 디자인 할 수 있다.

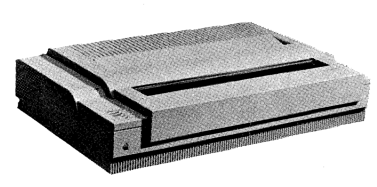
또한 이 제품은 영문의 ESC / P 사양은 물론 2바이트 완성형, 조합형, N바이트 한글 및 4,888자 한자의 KSSM과 정부표준인 KS 한글 / 한자를 지원, 원하는 모드대로 선택 및 사용이 가능하다.

특히 30여가지의 다양한 기능을 액정 디스플레이의 프론트 콘트롤 판넬을 통해 직접 조작, 확인이 가능하고, 속도향상에 비례한 소음에 있어서도 55dB로 소음도를 낮춰 사무실에서의 프린팅시에 쾌적하게 이용할 수 있다.

- 인자방식 : 도트매트릭스 방식
- 핀의 수 : 24핀
- 인자속도 : 영문 Draft–333CPS,
  한글 보통–74CPS / 고속–148CPS
- 인쇄크기 : 영문–136글자 / 줄(보통),
  233글자 / 줄(축소)
  한글–102글자 / 줄(보통),
  204글자 / 줄(축소),
- 문자체 : 영문–Draft, Roman, Sans Serif
  한글–명조체, 고딕체
  한자–명조체
- 특징 : 흑백은 물론 7가지 칼라 프린팅 가능

문의처 : 삼보컴퓨터
전 화 : (02) 313–0030

봉투 등도 인쇄 가능한
Apple Image Writer LQ



이 제품은 각종 형태의 프린트 작업에 적합한 다용도의 도트 매트릭스 프린터로 고품위글자, 그래픽 뿐만 아니라 Paper Handling, 7가지 색상제공 및 네트워크기능까지 제공한다.

또한 매킨토시 뿐만 아니라 기존의 애플II와 완벽한 호환성을 유지하고 있으며 다양한 Paper Handling 기능들

–Single Push / Pull Tractor, Bottom Feeder, Cut Sheet Feeder(선택)에 의해 두꺼운 지질, 봉투등에도 손쉽게 프린트할 수 있게 되었다.

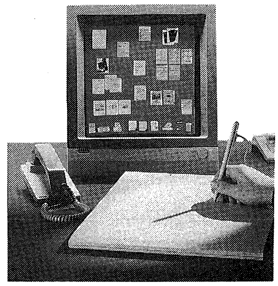
주요 특징을 보면 다음과 같다.

- 프린트 방식 : 27핀 도트 매트릭스
- 프린트 모드 / 속도
  Draft : 10CPI에서 250CPS
  NQL : 10CPI에서 90CPS, 144DPI에서 140CPS
  LQ : 216DPI에서 115CPS
- Mac용 자형 및 사이즈 : Time, Helvetica Courier, Symbol 등의 범용자형, Bold, SuperScript, Underline, Proportional Text등
- Paper Handling : Letter Head, 라벨, 봉투, 두꺼운 종이 사용가능
- 복사 매수 : 5장 동시 인쇄
- 용지 하향 공급기능 : 다양한 형태의 Form지 및 라벨, 두꺼운 재질의 종이등을 공급, Jam방지 기능
- Apple Talk(선택) : 31명의 유저가 동시 공유
- 칼라기능 : 칼라리본 이용 4가지 색상가능
- S / W 호환 : 대다수의 Apple II 및 Mac family간의 호환성 제공
- 캐릭터 Pitches : ROM의 내부폰트 이용

문의처 : (주)엘렉스컴퓨터
전 화 : (02)780-4545

사무자동화 패키지
왕 프리스타일 시스템

왕 프리스타일 시스템은 왕의 PC 200 시리즈 뿐 아니라 산업표준 PC에서 사용할 수 있는 하드웨어와 소프트웨어가 결합된 제품으로 전자펜, 전자판, 전화기 세트를 이용하여 쓰거나 말을 함으로써 컴퓨터 스크린에 정보를 입력, 관리하고 타 시스템 사용자와 통신할 수 있다.

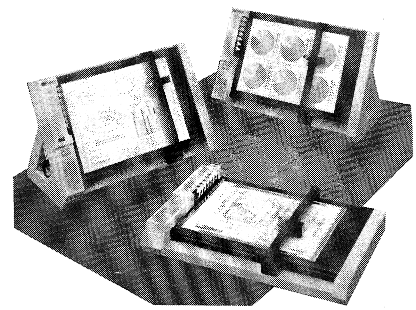


동 제품의 특징은 전자펜으로 전자판을 터치하면 PC 스크린이 같은 크기의 전자종이로 변하여 그 위에 전자펜을 이용하여 사인이나 글자를 쓸 수 있고 전화기 세트를 이용하여 그 문서에 대한 코멘트를 음성으로 합성시킬 수 있다. 또한 실제 사무실의 책상위와 같이 사무기기가 놓여 있는 데스크의 모형이 나타나는 전자 책상에서 전자판위에 펜을 움직여 스크린에 나타난 모형을 지정하여 컴퓨터에서 서류의 묶음, 파일, 발송, 실제 크기로의 확대 등의 명령을 줄 수 있다.

- 프리스타일의 기본장비
  PC 200 / 300 시리즈 CPU : 고해상도 모니터 모델 1640과 콘트롤러와 표준해상도 모니터 1240과 허클레스 그래픽 호환 비디오 카드를 지원 전자판과 테블릿 인터페이스 카드, 프리스타일 소프트웨어
- 음성합성장비
  프리스타일 보이스 인터페이스 카드, 프리스타일 보이스 소프트웨어
- 프린터 연결사용
  LCS-15 또는 LDP-8 프린터와 소프트웨어
- VS 오피스 네트워크 연결사용
  WLOC 카드, VS / 프리스타일 인터페이스 소프트웨어, VS OS 7.13 이상, FAX 인터페이스 카드, 프리스타일 FAX 소프트웨어
- WIIS 시스템과 연결 사용
  WLOC 카드, VS / 프리스타일 인터페이스 소프트웨어
- 스캐너 사용
  SC300 스캐너, SC300 인터페이스 카드, 프리스타일 스캐너

문의처 : 왕컴퓨터코리아(주)
전 화 : (02)784-6111

데스크탑형 플로터
DXY시리즈



이 DXY시리즈(1100, 1200, 1300)는 마이크로 스텝핑모터를 사용하여 0.0125㎜ / step의 해상도와 420㎜ / s의 고속 플로팅이 가능한 데스크탑형 플로터이다.

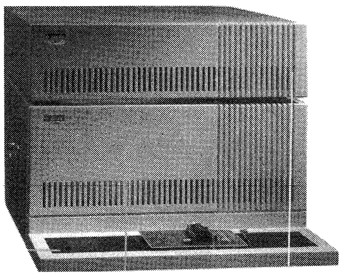
이중 DXY-1300은 1M바이트의 버퍼가 내장되어 있어 플로팅중에도 컴퓨터가 에디팅할 수 있으며, RD-GL I코맨드와 HP-GL코맨드를 내장하여 Auto CAD, Micro CAD / CAM, P-CAD 등 현재 사용되고 있는 대부분의 CAD / CAM / CAE소프트웨어를 지원할 수 있다. 펜 제어는 소프트 랜딩(Soft Landing) 메카니즘방식을 채택하고 있어 플로터의 수명을 연장시켜 주고 있다.

주요 규격을 다음과 같다.

- 용지크기 : A3
- 펜의 종류 : 잉크펜, 세라믹펜, 볼펜, 파이버팁펜
- 사용되는 펜의 수 : 8펜
- 기계적인 해상도 : 0.0125㎜ / step
- 버퍼 : 1MB(옵션 1MB RAM접속 가능)
- 반복성(Repeatability) : 0.1㎜이하

문의처 : (주)태일시스템
전 화 : (02)756-5340

ASIC 분석시스템
SYSTEM 6000



이 제품은 ASIC제품을 테스트하여 줄 뿐만 아니라 그 내부기능까지도 분석하여 주는 애널리시스 시스템이다. 즉, 로직 애널라이저와 VLSI 테스터의 성능을 모두 갖추고 있어서 단순한 ASIC 테스터는 물론 ASIC를 포함한 보드레벨이나 시스템레벨의 기능 및 타이밍 분석 등 모든 레벨에서의 디바이스를 분석할 수 있는 시스템이다.

또한 이 제품은 테스터 과정에서 셋업(Set Up)할 경우, 각각의 개별카드를 만들지 않아도 소프트웨어로 지정이 가능하기 때문에 시간과 경비를 줄일 수 있다. 그리고 자동 벡터발생기에 의해 테스트 벡터를 쉽게 만들 수 있고 테스트 벡터의 변환이 용이하도록 설계되어 있어서 시뮬레이터 출력을 바로 들일 수 있다.

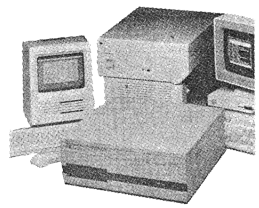
더우기 SYSTEM 6000은 사용자가 쉽게 알 수 있도록 메뉴 드라이브(Menu Drive) 방식으로 설계되어 있고 크기 또한 작아서 데스크탑 PC와 같이 사용할 수 있으며, 기존 ATE장비에 비해 훨씬 가격이 저렴하다.

그 밖에 주요 규격은 다음과 같다.

- 테스트 핀 : 최대 352핀까지 가능하며, 전 핀은 입력이나 출력 혹은 양쪽 모두 지정할 수 있다.
- 테스트 속도 :
  클럭 속도-50MHz까지 가능하다.
  데이터나 애퀴지션 속도-25MHz까지 가능하다.
- 분해능 : 25mV
- 비교 스트로브 범위 : 0~1000(1ns)

문의처 : (주)한국실리콘
전 화 : (02)783-7171~3

토탈네트워킹 미니급 컴퓨터
3S/400

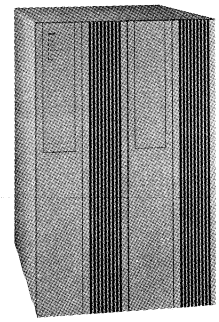


본 시스템은 토탈네트워킹 시스템으로 이더네트(Ethernet), 애플토크(AppleTalk), 토큰링(Tokenring) 등의 사용이 가능하며, 마이크로에서 미니, 메인프레임 등의 어떠한 아키텍처와도 연결이 가능하다. 또한 멀티태스킹의 완벽한 구현을 할 수 있고 많은 PC용 소프트웨어 패키지를 DOS하에서 운용하면서 LAN의 문제점인 한글데이터의 전송도 가능하다. 그리고 워크스테이션용으로 1,024대까지 연결이 가능하다.

- CPU : 80386, 32비트(16MHz)
- Co-Processor : 82586
- 메모리 : 기본 2MB, 14MB까지 확장 가능
- 보조기억장치 : 기본 : 150MB
  최대 : 900MB까지 확장 가능
- 네트워크 인터페이스
  -이더네트(Ethernet) 및 AppleTalk
  -토큰링
- 페러럴 포트 및 슬롯
  - 4개의 16비트 AT 확장슬롯
  - 2개의 32비트 메모리 확장슬롯
  - 3개의 RS-232C 시리얼포트
  - 1개의 패러럴포트
- OS : MS-DOS, OS / 2
- 터미널기능 : 최대 1,024대

문의처 : 컴퓨터월드
전 화 : (02)332-0141

32비트 멀티 프로세서 시스템
NCR TOWER 32/800



이 시스템은 멀티유저 범용시스템인 NCR TOWER 32 시리즈 중 최신의 모델로서 화일 프로세서(FP), 터미널 프로세서(TP), 클러스터 콘트롤러 호스트 프로세서(CCHP), 통신 프로세서(CP), LAN 프로세서(LP), 멀티버스 I 인터페이스 프로세서(IP) 등 최고 6개의 68020 어플리케이션 프로세서(16.67MHz)까지 가능하도록 설계되었다.

주요 특징을 보면 다음과 같다.

- CPU : (6개의) 68020
- Co-processor : 68881
- 속도 : 각 프로세서당 16.67MHz
- RAM : 각 프로세서당 4MB(16MB 확장가능)
- 보조기억장치 : 170MB 또는 380MB의 ESDI 디스크드라이브(5.25인치) / 150MB SCSI 카트리지 테입 드라이브
- 표준 OS : UNIX시스템 V
- 터미널 : 256유저 사용 가능
- 슬롯 : 20
- 중량 : 180kg
- 전압 : 180~257VAC, 50 / 60Hz

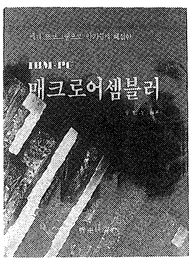
문의처 : 동아컴퓨터(주)
전 화 : (02)784-6821

● PC레이다 ●

신간안내

IBM-PC 매크로 어셈블러

신동준 편저/기전연구사 발행/
정가 6,000원/348면/4·6배판



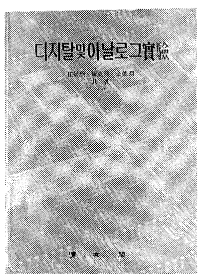
어셈블리 언어에 대해 많은 사람들이 관심을 갖고 공부를 하고 있지만 본격적으로 프로그램을 짤수있는 단계까지는 꽤 많은 시간이 소요된다. 그만큼 프로그래밍에 대한 개념은 익히기가 어렵다. 뿐만 아니라 어셈블리 프로그램을 작성하는 수단인 매크로 어셈블리에 대해 알기 쉽게 설명한 서적도 별로 없다.

본서는 이러한 어려움을 겪고 있는 이들을 위해 간단한 예제 프로그램을 제시하는 방법을 사용하여 어셈블리 프로그래밍의 개념과 매크로 어셈블러의 본격적인 활용방법에 대해 알기 쉽게 설명되어 있다.

본서는 모두 5장으로 구성되어 있다. 1장에서는 매크로 명령의 사용법에 관하여 설명함으로써 효율적인 프로그래밍 기법을 체득할 수 있도록 되어 있다. 제2장에서는 화일 입출력 방법을 설명한다. 3장과 4장에서는 모듈화기법 및 라이브러리 기법에 대하여, 5장에서는 고급언어 프로그램과 어셈블리 루틴을 결합하는 방법에 관하여 설명돼 있다.

디지탈 및 아날로그 실험

최호열외 2인 공저/청문각 발행/
정가 7,500원/496면/4·6배판



대부분의 실험은 학문적 이론을 실제적으로 증명하는데 그 의의를 두고 있기 때문에 실험을 실행하기 전에 우선적으로 실행하고자 하는 내용에 대해 이론적으로 철저히 이해하는 과정이 필요하다.

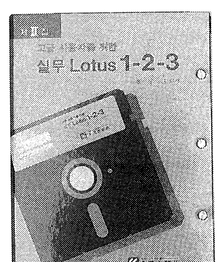
따라서 본서는 전기, 전자, 전자계산기 공학 및 전자통신분야를 전공하는 학생들이 실험하기에 앞서 필수적으로 이해해야 되는 기본이론을 상세하게 풀이하였다. 또한 응용력을 키울 수 있는 과제들을 각 실험마다 다루었다.

그러나 실험은 이론의 실증이외에도 새로운 것에 대한 탐구나 기술습득에도 그 의의를 두고 있기 때문에 실험결과에 대한 철저한 분석과 고찰이 반드시 들어가 주어야 한다. 이에 본서에서는 각 단원 끝부분에 실험결과에 대한 고찰을 체계적으로 할 수 있도록 문제가 제시되어 있다.

본서는 TTL IC와 조합논리회로, 플립플롭, 각종 카운터를 중심으로 한 순서논리회로, CMOS IC, 각종 인터페이스 회로, 클럭신호발생장치, 반도체 메모리회로 및 A–D / D–A변환, 그리고 OP앰프와 각종 필터, PLL(Phase Locked Loop)에 대한 이론과 응용회로의 구성법에 대해 다루고 있다.

실무 Lotus 1-2-3

홍수현 편저/영진출판사 발행/
정가 12,000원/254면/8절판



본서는 Lotus 1–2–3의 기본적인 활용방법을 알고 있는 이로서 전문적인 업무에 관련된 표를 작성하고자 할 때 유용한 책이다.

막연히 추상적인 개념만을 근거로 컴퓨터를 이용하여 작성된 표를 실제로 이용했을 경우, 일반적으로 많은 허점이 발견된다. 따라서 그러한 결함을 보완하기 위해 작성자는 여러번 손을 보는 불편함을 겪게 된다.

이럴때 실제 사용할 수 있는 예제가 있다면, 이를 참고로 작성자 실정에 맞게 수정하거나 작성할 수 있어 시간과 노력을 절약할 수 있을 뿐 아니라 새로운 표를 작성할 때 유사한 테크닉을 발휘할 수 있을 것이다.

이러한 것을 염두에 두고 본서는 재무회계관리, 도매업제품 출고관리 등에 대한 프로그램을 예제로 수록하였기 때문에 이를 토대로 독자들이 자신의 업무에 맞게 변형시키면 간단히 전산화를 이룩할 수 있을 것이다.

그리고 본서의 내용은 Lotus 1–2–3 뿐만 아니라 Quattro와 같이 Lotus 1–2–3과 호환성있는 프로그램에서도 똑같이 사용이 가능하도록 되어 있다.

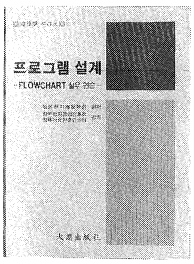
Lotus 1–2–3의 활용방법을 잘 모르는 독자의 경우도 자신이 필요로 하는 표와 유사한 예제를 찾아 그 내용을 파악하고 함수나 매크로 등이 어떻게 사용되었나를 이해하면 많은 도움이 될 것이다.

또한 독자들이 실무에 참고할 수 있도록 소프트웨어들은 디스켓으로 따로 공급하고 있다.

프로그램 설계
–플로우차트 실무연습–

한국전자공업진흥회 컴퓨터요원 훈련센터 편역/대은출판사 발행/
정가 5,000원/250면/4·6배판

컴퓨터 기술의 다양화는 고도의 프로그래밍 기술을 필요로 하고 있다. 그러나 컴퓨터 이용기술이나 프로그래밍 기술은 전산화의 완벽한 계획과 구성에 의해 이루어지므로 프로그램 설계의 비중이 가중되고 있다.

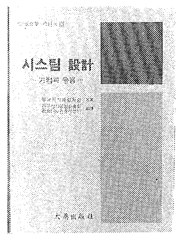


여기서 프로그램 설계는 프로그램을 작성하기 전에 부분적으로 어떤 기능을 어떻게 실행할 것인가를 고려하여 전체적으로 그 구상들을 통합하는 과정이다. 본서에서는 이러한 프로그램 설계과정을 가장 논리적인 표현이 가능한 플로우차트를 중심으로 설명하였는데 플로우차트와 화일의 기본적인 개념부터 설명되어 있어 기본개념이 없는 독자들도 볼 수 있도록 되어 있다. 특히 일반적으로 작성되는 프로그램의 과반수 이상이 사무처리용이기 때문에 사무처리용 프로그램 언어인 코볼을 기초로 하였다.

또한 이들은 플로우차트 작성의 기본적인 세가지 패턴인 직선형, 분기형, 반복형 등을 잘 구성하여 설명되어 있고, 각장마다 구체적인 사무처리 분야의 예제를 제시하였으며, 독자들의 응용력 향상을 목적으로 직장별로 주요 부분을 재정의하여 복습할 수 있도록 되어 있다.

시스템 설계
-기법과 응용-

한국전자공업진흥회 컴퓨터요원 훈련센터 편역/대은출판사 발행/
정가 5,000원/246면/4·6배판



시스템 분석과정은 전산화에 있어서 약 60%의 비중을 차지하고 있다해도 과언이 아니다. 이에 따라 시스템 설계기법이 중요시 되는 것도 당연하다 하겠다.

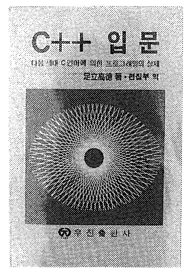
본서는 이러한 시스템 설계기법을 실무문제 해결에 중점을 두어 좀더 체계적으로 습득할 수 있도록 한 책으로서, 설계순서의 표준화 및 설계결과의 문서화를 기하였고 배치처리를 중심으로 해설하였다.

또한 본서는 풍부한 문제를 수록하여 독자들이 좀더 심도있게 이해할 수 있도록 하였으며, 부록으로 소규모 시스템의 설계 연습을 함으로써 종합적인 이해를 돕는 등 시스템의 설계기법뿐 아니라 응용할 수 있는 토대까지도 마련해준다.

C++입문

편집부 역/우진출판사 발행/
정가 4,000원/191면/국판

C++는 C언어의 계승언어로서 미국 벨연구소에서 개발한 범용프로그래밍 언어이다. 즉



C++ 는 C의 축적된 소프트웨어 기술을 기초로 만들어진 「새로운 C」이다.

C는 언어형식이 작기 때문에 퍼스컴과 같은 작은 시스템에도 비교적 간단히 설치할 수 있고, 어셈블러 등에 비하면 고급언어이면서 하드웨어에 밀착된 조작도 간단히 기술할 수 있기 때문에 재빠르고 분명한 프로그램을 쓸수 있어 시스템 프로그램에서 응용프로그램까지 폭넓게 사용되고 있다.

C++는 C로 기술하는 것을 모두 기술할 수 있는 것은 당연하고, 그외에도 오브젝트 지향 프로그래밍을 실천할 수 있다.

본서는 이러한 C++에 의해 프로그래밍을 해설한 입문서이다. 1장에서 3장까지는 주로 초보자 즉 MS-DOS나 에디터의 사용법은 알고 있으나 프로그램은 처음인 독자들을 위해 프로그래밍의 기본적인 요소를 해설하였다. 그러나 이 부분에서 C를 알고 있는 독자들은 C와 C++의 미묘한 차이점을 익힐 수 있을 것이다.

후반부인 4장에서 9장까지는 C++의 키워드인 「클라스」라는 개념을 사용한 프로그램 개발에 대해 설명하고 있다.

또한 본서는 실제적인 문제를 해결하기 위한 프로그램을 조금씩 만들어 가는 과정에서 필요에 따라 새로운 기능을 배워갈 수 있는 방법으로 편집되어 있다.

OS/2 PS/2 철저 분석

김영민·김원희 공저/컴퓨터월드 발행/
정가 9,000원/366면/신국판



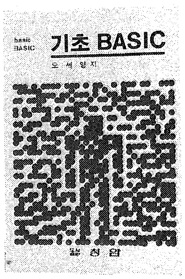
IBM-PS / 2는 퍼스컴의 위치를 메인프레임, 미니컴퓨터 등의 자원을 활용하는 하나의 패밀리로 설정지웠다는 점에서 큰 의미가 있다.

본서는 이러한 IBM OS / 2의 제2혁명이라 불리는 PS / 2와 그 운영체계인 OS / 2에 대한 완전한 해설서로 퍼스컴의 새로운 운영체계로 주목받고 있는 OS / 2의 기능해설과 함께 PS / 2 각 모델별 주요 특징과 규격이 상세히 설명되어 있다.

특히 이 책은 IBM PS / 2 발표이후, 시장질서가 새롭게 재편돼 가고 있는 퍼스널컴퓨터 업계에 있어서 국내 호환기업체들의 공동 대응책마련을 위한 기술지침서가 될 것이다.

기초 BASIC

오세영 저/청암출판사 발행/
정가 4,500원/320면/국판



컴퓨터는 이제 우리 생활에 점차 뿌리를 내리고 있고, 그에 대한 의존도 또한 날로 증대하고 있는 실정이다. 여기에 컴퓨터를 이용해서 문제를 해결하고 계산하고 정보를 처리하기 위해서는 프로그래밍 언어로 프로그램을 작성하지 않으면 안된다.

그리하여 본서에서는 베이직 언어에 관한 개념이나 기술 흐름도작성의 기본사항, 프로그램의 논리 등 간결하고 알기 쉽게 단계적으로 자세히 설명하ㄱ 있어 많은 도움이 되리라 본

DISK OPERATING
SYSTEM 입문

APPLE

성평식 편저/청암출판사 발행/
정가 4,500원/229면/4·6배판



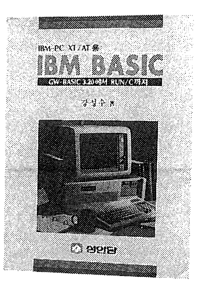
퍼스컴의 보급이 활발히 진행됨에 따라 각사의 여러 제품이 시중에 나와 있다. 그런데 퍼스컴의 탄생기부터 지금까지 명맥을 이어오는 퍼스컴으로 APPLE Ⅱ가 있다. 이 APPLE Ⅱ는 하드웨어적인 고도한 기능을 가진 동시에 APPLE Ⅱ용에 맞는 다양한 소프트웨어가 개발되어 있다.

본서는 APPLE Ⅱ의 디스크 이용에 관한 초보자를 위한 것으로 버전 DOS3.3을 기준으로 하여 서술하였다. 또한 DOS3.2 이하의 버전과의 차이점에 대해서도 서술하고 있으며 본서를 참고함으로써 APPLE DOS의 이해가 쉽게 되리라 믿는다.

IBM-PC XT/AT용

IBM BASIC

강성수 저/성안당 발행/
정가 6,000원/376면/4·6배판



퍼스널컴퓨터가 우리들 일상생활의 영역에까지 보급되면서 그 프로그램의 중요성 또한 날로 증대해 가고 있다. 이 컴퓨터를 활용할 수 있는 프로그램 언어가 많이 있지만 그 중에서도 특히 베이직언어는 초보자라도 쉽게 익힐 수 있는 편리한 언어로서 이의 활용이 활발하다.

본서는 국내외에 많이 보급되어 있는 IBM 퍼스널컴퓨터를 중심으로 한 GWBASIC 3.20언어를 기초부터 쉽게 익힐 수 있도록 단계적으로 상세히 설명한 후, 이 이론을 바탕으로 엄선한 예문을 풀어 가는 동안에 어려운 응용 문제의 프로그램도 개발해 낼 수 있는 단계에까지 이르도록 전개하고 있다.

또한 그래픽 명령이나 베이직과 유사한 RUN/C 언어에 대해서도 설명하고 있다.

초보자를 위한 가이드

IBM PC 입문

탁연상 저/영진출판사 발행/
정가 4,000원/210면/4·6배판



개인용컴퓨터의 급속한 보급으로 PC는 개인의 취미나 학습용 뿐만 아니라 회사의 업무처리에서도 활발히 이용되고 있다.

본서는 컴퓨터에 대한 전문지식이 없는 사람들이 컴퓨터에 두려움 없이 접근할 수 있도록 PC의 구입에서 설치하는 단계, 디스크의 사용법이라든가, 베이직언어에 대해서도 자세한 설명을 하고 있다.

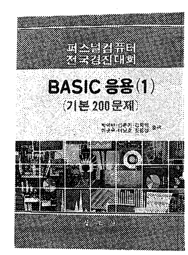
또한 응용소프트웨어 패키지, 즉 워드프로세서, 데이터베이스, 스프레드쉬트 등에 관해서도 서술하고 있다.

컴퓨터를 배우고자 하는 초보자들에게는 좋은 안내서가 되리라 본다.

퍼스널컴퓨터 전국경진대회

BASIC 응용(1)

박덕만외 3인 출제/상조사 발행/
정가 3,500원/256면/4·6배판



오늘날 컴퓨터의 눈부신 발전는 제2의 산업사회라 할 수 있는 정보화사회를 탄생시키는 데 주요한 역할을 해왔다.

따라서 본서는 컴퓨터교육과 마인드의 확산을 위한 정부의 의지의 하나인 컴퓨터 경진대회를 대비할수 있도록 구성하였다.

우선 기본적인 명령을 확인한 후 기본적인 패턴문제 200제를 상세한 해설로 이해할 수 있도록 하였다. 또한 각 패턴문제에 대한 예제문제가 수록되어 있어 스스로 문제 이해능력을 파악하고, 나아가서 패턴문제를 응용할 수 있도록 정선문제를 수록하였다.

부록에는 전국대회 실기문제와 오류메시지를 첨부하였다.

첨단서적

컴퓨터 전문서적 통신판매 써비스

도서명	출판사	가격
표준 MS DOS 핸드북	기전연구사	7,000원
MS DOS 핸드북	정보시대	3,000원
MS DOS 입문	성안당	5,000원
MS DOS	성안당	5,000원
IBM DOS 3.2	크라운	5,000원
MS DOS 3.3	크라운	6,000원
알기 쉬운 MS DOS	영진	5,000원
고급과정 MS DOS	크라운	6,000원
MS DOS 핸드북	크라운	9,000원
DOS 3.3	영진	8,000원
MS DOS 4.0	연학사	6,500원
MS DOS 메뉴얼	크라운	4,000원
MS DOS와 GW BASIC	연학사	5,500원
Lotus 123 기본	크라운	5,000원
실무 Lotus	크라운	6,000원
Lotus Leal 2	크라운	5,000원
완벽 Lotus	영진	5,000원
알기 쉬운 Lotus	크라운	9,000원
Lotus 123	기전연구사	6,000원
실무 Lotus 123	영진	10,000원
실무 Lotus 123(제2장)	영진	12,000원
dBASE Ⅱ	크라운	4,000원
dBASE Ⅲ	기전연구사	5,500원
실무 dBASE Ⅲ plus	영진	6,000원
초보자를 위한 dBASE Ⅲ plus	영진	6,000원
프로그래머를 위한 dBASE Ⅲ plus	영진	6,000원

도서명	출판사	가격
알기 쉬운 dBASE Ⅲ	영진	6,000원
dBASE Ⅲ plus 프로그램 모음집	영진	9,000원
dBASE Ⅲ plus	크라운	7,000원
dBASE Ⅲ plus	기전연구사	6,000원
Auto CAD 2.6	영진	15,000원
Auto CAD 실무	크라운	5,500원
Auto CAD 실습과 응용	영진	10,000원
C언어 기초+α	교학사	8,000원
Run C	교학사	7,000원
C언어의 이론 응용	성안당	7,000원
C 프로그래밍	동일	5,000원
C언어 500제	크라운	5,000원
C언어 입문과 실습	크라운	5,500원
C 라이브러리	영진	5,000원
TURBO C(User's)	크라운	5,000원
TURBO C(Reference)	크라운	5,000원
MS C 교과서	성안당	6,000원
Clipper Compiler	영진	5,000원
실무자를 위한 Crosstalk	영진	5,500원
노턴 유틸리티	영진	3,500원
노턴 유틸리티	크라운	4,000원
XENIX V 메뉴얼	영진	7,000원
XENIX 오퍼레이팅 시스템	크라운	5,500원
Dr. HALO Ⅱ	교학사	3,000원
Dr. HALO Ⅲ	교학사	3,000원
센서 인터페이싱	기전연구사	24,000원

도서명	출판사	가격
SIDEKICK plus	성안당	6,000원
GW BASIC 3.2	성안당	4,500원
GW BASIC	크라운	5,500원
IBM BASIC	성안당	6,000원
워드 퍼펙트	우진출판사	7,000원
PC TOOLS DELUXE	크라운	3,000원
터보파스칼 4.0	크라운	8,000원
WORDSTAR 4.0	우진	6,000원
WORDSTAR 실무	영진	4,500원
애플 Ⅱ 입문	영진	3,500원
Symphony 메뉴얼	크라운	7,000원
Symphony 메뉴얼	영진	7,000원
RM COBOL	성안당	5,000원
MS COBOL	성안당	6,000원
IBM PC 어셈블리	기전연구사	6,000원
IBM PC 하드웨어 종합분석	성안당	5,500원
MSX DOS 입문	연학사	5,500원
MSX DOS 슈퍼 핸드북	우진	4,500원
MSX BASIC 컴퓨터교실	기전연구사	4,000원
컴퓨터 용어사전	영진	4,500원
컴퓨터 용어사전	대은	9,000원
미니탭 메뉴얼	크라운	6,000원
APPLE Ⅱ 종합분석	성안당	5,000원
APPLE BASIC 프로그램 실습	영진	6,000원
인공지능	영진	4,000원
마이크로프로세서	동일	7,000원

※ 본지에 게재된 기타 신간서적도 주문 가능함

- **주문방법** : 직접 구입 또는 교재명과 출판사명을 기재하여 해당금액에 발송료 500원을 추가하여 우체국 소액환으로 동봉 신청하면 즉시 우송함.
- **보 낼 곳** : ㉾140-012 서울시 용산구 한강로2가 전자단지 17동 3층 가열 325호
  전화 : 716-6024

전문인을 섬기는 **첨 단 서 적**

THE PROGRAMMER'S SHOP

helps save time, money, and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

C Talk by CNS - Object-oriented C programming. Smalltalk-like message formats contains encapsulation, messaging, inheritance. MS, Lattice, Turbo, C86. MS $ 129

386 Development Tools

386 Assembler/Linker	PC	$ 379
386 Debug - by Phar Lap	PC	$ 139
386/DOS Extender	PC	$1395
DESQview PS/2	PC	$ 109
F77-LEM/32 bit - by Lahey	MS	$ 869
OS/286 & 386 by AI Architects	PC	Call
Paradox 386	MS	$ 799
PC-MOS/386	PC	$ 169
Windows/386	MS	Call

AI Expert System Dev't

1st Class	MS	$ 399
Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
CxPERT V3.0 - shell for C	MS	$ 349
Experteach II	PC	$ 339
Exsys	PC	$ 289
Level 5 - formerly Insight 2	MS	$ 589
T.I.: PC Easy	PC	$ 419
Personal Consultant Plus	PC	$2589
Turbo Expert-Startup(400 rules)	PC	$ 119
Corporate (4000 rules)	PC	$ 339

AI-Lisp

PC Scheme LISP - by TI	PC	$ 77
Star Sapphire	MS	$ 429
TransLISP - learn fast	MS	$ 79
TransLISP PLUS	MS	$ 139

Others: IQ LISP ($155), IQC LISP ($269)

AI-Prolog

APT - Active Prolog Tutor	PC	$ 49
ARITY Prolog - Interpreter	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
Cogent Prolog Compiler	MS	$ 179
MicroProlog Prof. Comp./Inter.	MS	$ 609
TURBO PROLOG by Borland	PC	$ 69
Turbo Prolog Toolbox	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS_PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 79
db/Lib	PC	$ 119
Exim Toolkit - full	PC	$ 85
Finally - by Komputerwerk	PC	$ 85
Inside Track	PC	$ 45
Mach 2 by MicroHelp	PC	$ 55
NetWurkz by Exim	PC	$ 89
Peeks 'n Pokes	PC	$ 30
QBase	MS	$ 79
QuickBASIC	MS	$ 63

FEATURES

CO/SESSION - Remotely: Access PC, peripherals, train or trouble-shoot. Session record/playback, file transfer keyboard and voice modes MS $ 229

CLARION DBMS by Barrington Systems. Fast applications prototyping and development. Language, compiler, screen/report generators, editor, call other languages, read/write dBASEIII+ files. PC Call

LIFEBOAT INC.

678-39, Yuksam-Dong, Kangnam-Ku, Seoul, Korea

PHONE : (02)555-1392

FAX : (02)555-9250

Basic

Quick Pak-by Crescent Software	PC	$ 59
Quick-Tools by BC Associates	PC	$ 109
True BASIC	PC	$ 79
Turbo BASIC - by Borland	PC	$ 69
Turbo BASIC Database Toolbox	MS	$ 69

C Language-Compilers

AZTEC C86 - Commercial	PC	$ 499
C86 PLUS - by CI	MS	$ 359
Datalight Optimum-C	MS	$ 109
High C Optimizing Compiler	PC	Call
Instant-C - source, debug.	MS	$ 369
Instant-C/16M	PC	Call
Lattice C - new version	MS	$ 249
Microsoft C 5.0 - Codeview	MS	$ 299
Microsoft Quick C	MS	$ 69
Rex-C/86-standalone ROM	MS	$ 695
Turbo C by Borland	PC	$ 64

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$ 219
C Trainer - by Catalytix	PC	$ 89
Interactive C by IMPACC Assoc.	PC	$ 189
Run/C Professional	MS	$ 145
Run/C Lite	MS	$ 79
Turbo C-terp	PC	$ 119

C Libraries-Communications

Asynch Manager by Blaise	PC	$ 135
Essential Comm Library	PC	$ 125
With Breakout Debugger	PC	$ 189
Greenleaf Comm Library	PC	$ 129
Multi-Comm - add multitasking	PC	$ 135

C Libraries-General

Blackstar C Function Library	PC	$ 89
C Tools Plus - V5.0	PC	$ 99
C Utilities by Essential	PC	$ 119
Entelekon C Function Library	PC	$ 119
Greenleaf C Sampler	PC	$ 69
Greenleaf Functions	PC	$ 125
LIGHT TOOLS by Blaise	PC	$ 69
Turbo C Tools by Blaise	PC	$ 99

C Libraries-Files

BTree - Source, no royalties	MS	$ 69
CBTREE-Source, no royalties	MS	$ 129
ctree by Faircom-no royalties	MS	$ 309
ctree w/ rtree	MS	$ 519
rtree - report generation	PC	$ 239
dB2C Files	MS	$ 259
dB2C Toolkit V2.0	MS	$ 249
dbVISTA - Object only	MS	Call

RECENT DISCOVERY

DBug III by HJS Research - Source level Dbase debugger. Breakpoints, view variable values, stop execution when variable changes, multiple windows. Stop runaway programs. MS $ 159

C-Screens, Windows, Graphics

C Power Windows by Entelekon	PC	$ 129
C Worthy Interface Library	PC	$ 165
Curses by Aspen Scientific	PC	$ 109
dBASE Graphics for C	PC	$ 69
ESSENTIAL GRAPHICS-fast	PC	$ 185
Font WINDOW/PLUS	PC	$ 229
GraphiC - new color version	PC	$ 279
Greenleaf Data Windows	PC	$ 155
w/source	PC	$ 259
LightWINDOWS/C for Datalight	PC	$ 79
Panel/TC - for Turbo C	PC	$ 95
PC Forms-by Golden	MS	$ 119
ScreenStar - with source	PC	$ 169
Terminal Mapping System	PC	$ 279
TurboWINDOW/C-for Turbo C	PC	$ 75
View Manager - by Blaise	PC	$ 199
Vitamin C - source, menus	PC	$ 155
VC Screen - screen paint	PC	$ 129
Windows for C - fast	PC	Call
Windows for Data - validation	PC	Call
ZView - screen generator	MS	$ 149

C Support-Systems

Advantage C++	PC	$ 479
cENGLISH SK	MS	$ 89
C Sharp - **realtime**, tasks.	PC	$ 479
C ToolSet DIFF, xref, source	MS	$ 79
Lattice Text Utilities	MS	$ 89
Multi-C - multitasking	PC	$ 135
PC LINT-Checker, Amiga $89	MS	$ 99
PforCe++	PC	$ 219
Pre-C - Lint-Like	MS	$ 169
Quickshell - script compiler	PC	$ 349
/*resident-C*/ with source	PC	$ 159
Sapiens V8 - 8M workspace	PC	$ 269
SECURITY LIB - Source $199	PC	$ 115
Timeslicer	PC	$ 265

Cobol

Cobol spII by Flexus	PC	$ 329
Microsoft COBOL	MS	Call
PCDT by Procode	MS	$ 899
Realia - very fast	MS	$ 799
RM/Screens	PC	$ 339
RM/User + 5	MS	Call
Ryan McFarland COBOL	MS	$ 899
COBOL-85	MS	$ 899

DataBase & File Management

Advanced Revelation	PC	$ 779
CQL - Query System - source	PC	$ 359
DataFlex by Data Access	PC	$ 595
DataFlex multiuser	PC	$1049
Magic PC	PC	$ 169
Paradox - original	PC	$ 369
Paradox V2.0 List: $725	PC	$ 499

FEATURE

CLEAR by Clear Software - Clarify Logic for Enhancement, Analysis, and Review. Graphical flow of control for dBASE. HP $135 MS $ 89

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

뭐든지 물어보세요

랩탑과 일반 컴퓨터와의 차이점은

저는 3년째 Apple Ⅱ+를 사용하고 있읍니다. 앞으로 IBM PC를 구입할 예정이어서 여러 정보를 알아보다 랩탑(Laptop) 컴퓨터에 대해 언듯 들었읍니다. 랩탑컴퓨터가 일반 컴퓨터와 다른 점이 무엇인지 알려주십시요.

윤상건 / 서울 종로구 중학동 58번지

말 그대로 휴대용 컴퓨터를 랩탑이라 합니다.

랩(Lap)이란 무릎이란 뜻입니다. 따라서 랩탑컴퓨터는 말그대로 무릎위에 놓고 쓸 수 있는 컴퓨터라고 할 수 있는데 쉽게 이야기하자면 휴대용 컴퓨터입니다.

즉, 일상적인 업무를 더욱 효율화하기 위해 개인레벨의 퍼스컴을 손가방만한 크기로 콤팩트하게 디자인하였기 때문에 활용성이 뛰어나고 가장 큰 매력은 휴대가 가능해 기동성이 있다는 것입니다. 그리고 일반 PC와 거의 동등한 기능을 갖고 있읍니다.

이러한 랩탑컴퓨터는 IBM이 처음으로 PC를 출하했던 지난 81년 이전에 이미 오스본사나 엡슨사 등에 의해 선보였으나 작은 공간에 하드웨어적인 기능들을 축소시키다보니 처리능력이 일부분야(예를 들어 저널리스트)에 한정될 뿐 아니라 메모리용량 및 스크린기술 등의 미흡으로 인해 그다지 인기를 끌지 못했읍니다.

그러나 현재는 그 처리능력이 일반 PC와 거의 동등하여 일반 유저들도 언제 어디서든지 랩탑을 한대 휴대하고 있으면 다른 시스템이나 호스트컴퓨터와 인터페이스시켜 사용이 가능할 뿐 아니라 내부에 모뎀이 내장되어 있거나 내장할 수 있도록 되어 있어 데이터통신도 가능합니다.

전원은 일반 PC의 경우는 AC 외부전원만을 사용하는 반면 랩탑에서는 외부전원외에 배터리도 전원으로 사용이 가능합니다. 따라서 배터리를 사용할 경우에는 장소에 구애받지 않고 사용할 수 있으나 수명이 제한되어 있으므로 고해상도를 갖는 플라즈마나 EL디스플레이를 갖춘 하드디스크를 사용하는 경우에는 전력소모량이 많아 장시간 이용할 수 없읍니다.

국내에 보급된 랩탑의 대부분은 RS–232C가 장착되어 있고 FDD는 3.5"만 사용가능합니다.

한글처리에 있어서는 일반 PC와 전혀 색다른 하드웨어로 구성되어야 하기 때문에 아직까지 국내에서는 한글처리가 가능한 랩탑이 없으나 최근 뉴트론에서 이를 개발한 것으로 알려지고 있읍니다.

가장 큰 단점은 가격이 고가인 것인데 보통 IBM PC / XT호환 랩탑의 경우에 XT기종보다 비싸고 AT보다는 싼 가격입니다.

국내에서는 청원시스템과 삼성HP, 뉴텍코리아, 북두컴퓨터, 창명실업 등에서 공급하고 있읍니다.

이를 구입하실 경우에 IBM PC와 호환성이 있는지 메모리용량은 어느 정도되는지 무게는 가벼운지, 또 모든 각도에서도 읽을 수 있는 풀사이즈의 스크린을 갖고 있는지, 배터리 사용이 가능한지 등을 검토하여야 합니다.

조합형과 완성형코드의 차이점은

IBM PC호환기종에서 쓰이고 있는 2 바이트 한글의 조합형과 완성형 및 정부 표준코드의 차이점과 문제점에 대해서 알고 싶습니다.

홍상준 / 충남 온양시 온천1동 242–5

완성형코드는 미리 완성된 문자에 하나의 코드값을 할당하는 방식

요즈음 널리 쓰이는 IBM PC호환기종에서는 2바이트 방식 한글을 쓰고 있읍니다. 원래 이 말은 8비트 애플기종에서 각 음소당 1바이트를 할당하던 N바이트 한글에 대응되는 표현으로써 한글 1문자를 무조건 2바이트를 이용하여 표현하는 코딩방식입니다. 즉 '한글'을 나타내는데 N바이트 방식으로는 ㅎ+ㅏ+ㄴ+ㄱ+ㅡ+ㄹ, 즉 6바이트가 필요하나 2바이트 방식에서는 4바

이트가 필요합니다.

2바이트 방식은 다시 조합형 코드와 완성형 코드로 분류되는데 이는 컴퓨터 내부에서 글자를 형성하는 방법에 따른 것입니다. 조합형 코드는 각 음소를 5비트씩 할당해서 하나의 글자를 형성하기 위해 이들 음소(초성, 중성, 종성)의 코드들로써 결합하는 방식이고 완성형 코드는 이같이 각 음소의 코드값으로 더하는 방식이 아니라 미리 완성된 하나의 문자에 하나의 코드값을 할당하는 방식입니다.

1987년 한국공업진흥청에서는 '컴퓨터 한글/한자 표준코드'로 이 2바이트 완성형 한글을 통일시켰읍니다. 그러나 이러한 조치는 한글코드를 통일시킨 점에서는 의미가 있으나 한글표현범위 문제에 있어서 크게 논란이 되어 버리고 말았읍니다. 즉 조합형 코드를 이용했을 때 표현가능한 문자수가 11,000자인데 비해 완성형 코드인 경우는 겨우 2,350자 밖에 표현할 수가 없기 때문입니다.

xyz-3D는 구체적으로 어떤 기능까지 가지고 있는지

얼마전에 xyz-3D프로그램을 접하게 되었읍니다. 이 xyz-3D가 3차원 그래픽용 프로그램이라는 것을 알고 있으나 구체적으로 어떤 기능을 가지고 있는지 궁금합니다.

김태형/서울 노원구 공릉1동 517번지 59호

xyz-3D는 그래픽 전용 프로그램으로 시뮬레이션까지 가능합니다.

xyz-3D는 1970년대 메인프레임용으로 개발되어 어떤 지형이나 물체의 3차원 모델의 실제 투시도를 만들어 내는 프로그램으로 지금은 IBM-PC용으로도 나와 있어 많은 분야에서 유용하게 사용되고 있읍니다.

즉, 지정된 관측점으로부터 물체를 투시한 후에 눈으로 볼 수 있는 모형을 만들어 낼 수 있는데 물체를 확대시키거나 물체자체를 돌려가면서 보려고 할 때 보고자 하는 면의 투시도가 그림으로 나타납니다. 이 때 먼거리에서 가까운 거리까지 관측점을 자유자재로 선택할 수 있읍니다.

또한 그림의 경계를 표시하기 위해 윈도우(Window)를 임의대로 설정할 수도 있고, 프로그램내에서 그림을 모두 포함하는 윈도우를 자동적으로 명령할 수도 있읍니다. 그림이 기울어질 경우에는 윈도우를 적당히 기울일 수도 있읍니다.

그리고 xyz-3D는 히든라인(Hidden line)을 완벽하게 없애 주고, 실제 투시도와 2점투시도, 직각투시도, 단면도 등을 그릴 수 있는데 색이나 선 굵기도 마음대로 정할 수 있읍니다.

또 다른 위치에 유저가 그린 그림과 똑같은 것을 복제할 수 있고, 좌우를 반대로 할 수도 있으며, 그림자진 부분을 표시할 수도 있읍니다. 그리고 특정한 구조를 따로 기억시켰다가 다른 그림을 그릴 때 그대로 사용할 수 있읍니다.

뿐만 아니라 3차원 문자도 표시가 가능해 원하는 글을 쓸 수도 있고, 입력시킨 것이 유효한지를 프로그램내에서 검토한 후 오류가 있을 때는 에러를 발생하여 주는 기능 등도 있읍니다.

컴퓨터를 이용한 통신방식에는 어떤 것들이 있는지

퍼스널컴퓨터를 이용한 통신방식에 대해 알고 싶습니다.

배정곤/서울시 강동구 성내동 미주APT 3동 307호

일반전화망을 이용한 것이 가장 많이 사용됩니다.

퍼스컴통신이란 어떤 수단에 의해 이미 퍼스컴에 입력되어 있거나, 입력된 후 가공처리된 정보, 혹은 현재 입력되고 있는 정보를 1차 통신망을 통하여 원격지의 다른 퍼스컴이나 호스트 컴퓨터에 송·수신하여 정보를 구하는 일체의 행위를 말합니다.

대표적인 통신방법에는 퍼스컴과 퍼스컴 사이를 연결하는 일반전화망(PSTN : Public Switched Telephone Network)이나 패킷 데이터교환망(PSDN : Packet Switched Data Network), 그리고 회선 데이터교환망(CSDN : Circuit Switched Data Network) 등이 있읍니다.

PSTN은 현존하는 1차 통신망으로 국내외를 막론하고 가장 넓은 지역에 설치된 퍼스컴통신으로 가장 많이 쓰이고 있으며 PSTN을 통해 퍼스컴통신을 행하기 위해서는 모뎀이 필수적으로 쓰입니다.

퍼스컴에서 전송하고자 하는 정보는 디지탈형태인데, PSTN은 아날로그 음성신호 전송을 위한 네트워크이므로, 이들 디지탈신호를 아날로그로 변환시켜주는 역할과 수신측에서 아날로그신호를 디지탈로 변환시켜주는 역할을 모뎀이 수행합니다.

PSDN은 데이터전송을 위해 만들어진 네트워크이며, 국내의 경우에는 데이콤의 DNS가 여기에 해당됩니다.

LSDN에 텔렉스망을 연결한 통신은 퍼스컴통신 자체가 목적이 아니라 퍼스컴이 텔렉스단말기의 기능을 대체하여 사용하는데 더 큰 목적이 있읍니다.

구입가이드

프린터 구입시 주의사항

개인용 컴퓨터가 널리 보급되면서 그의 대표적인 주변기기인 프린터도 그 보급도가 증가되고 있는 추세이다. 그러나 이 프린터는 고가의 제품이라는 특성을 가지고 있기 때문에 구입하게 될 때 신중함을 기하지 않고는 불품없는 고가의 장식품을 마련하게 될 것이다.

현재 레이저 프린터가 매우 큰 관심거리로 사람들의 호기심을 자극하고 있으나 이 글을 읽고 있는 독자의 대부분은 임팩트방식의 도트매트릭스 프린터가 가장 접하기 쉬우며 많이 보급되어 있으므로 여기서의 설명도 이 방식의 프린터에 관하여 설명하고자 한다.

처리업무에 적합한가

프린터를 구입하게 될때 고려할 사항중에 하나가 처리업무의 성격에 맞는 것을 선택해야 한다는 것이다. 매일 매일 처리해야 할 업무량이 많은 곳은 인쇄속도가 빠른 속도의 프린터가 필요하며 각종 보고서를 작성해야 하는 업무에서는 인자품질이 뛰어나야 하기 때문이다.

●알맞은 속도의 선택

속도는 구동모터의 성능과 연관을 가지면서 가격구성의 중요한 요소가 되므로 자신의 업무에 어느 정도의 속도가 필요한 것인가를 충분히 고려하며 최저의 속도가 어느정도 되는지 살펴야 한다.

이러한 면에서 볼 때 24핀의 도트매트릭스 프린터의 속도는 그 한계가 300 CPS 정도이기 때문에 그 이상의 속도에서는 헤드가 마모되거나 글자의 오차율이 커지는 문제점이 발생하게 된다. 이러한 문제점을 해결시켜 주는 것이 레이저 프린터인데 이 레이저 프린터는 고가의 제품이기는 하지만 엄청난 속도와, 저소음이라는 측면에서 비용의 난점을 충분히 해소시켜 주고 있다.

그러면 속도면에 관하여 어떤 것들을 점검해야 하는지 살펴보자. 속도는 CPS(Character Per Second), 캐리지 리턴(Carriage Return), 배속인자(Draft), 버퍼, 바텀피드(Bottom Feed), DPI (Dot Per Inch), 절선이동(Skip of Perforation), 탭(Tap)의 기능 등이 그 변수로 작용하므로 이러한 기능을 점검해야 한다.

CPS는 프린트 헤드장치가 문자를 빈칸없이 일렬로 찍어 나갈 때 낼 수 있는 초당 인자하는 문자의 갯수를 의미하고 이는 프린터 속도의 기준이 된다. 그러나 빈칸, 빈행, 폼피드 등을 조화시켜서 측정할 때의 속도는 달라진다.

그러므로 여기서 주의할 사항은 속도가 빠르다고 해서 반드시 좋은 것이 아니라는 점이다. 예를 들어 500 CPS 정도의 속도를 가진다고 해도 헤드에 무리를 가하면 손상을 입게 되고 또 속도가 빠르게 되면 그만큼 열을 받게되어 열을 식힐 시간이 필요하게 되어 프린팅이 중지되기도 한다. 그러므로 전체적인 인자속도를 좌우하지는 않는다.

또한 인자하는 문자폭에 관하여는 80컬럼, 136컬럼의 개념이 있는데 보통 1인치당 10문자의 문자폭을 기준으로 한다. 136컬럼은 풀사이즈를 모두 이용할 수 있다는 장점이 있어 대체로 업무용에 쓰이고, 80컬럼은 교육용으로 쓰이고 있다.

캐리지 리턴은 프린터의 가동부분이 제자리로 돌아오는 것을 말한다.

드래프트(Draft)모드는 문자를 한 도트씩 건너뛰며 프린트하는 것으로 문자의 품질보다는 속도를 낼 때 사용된다. LQ 모드에서보다 약 1.3~2배가 빨라진다.

버퍼는 프린트 속도보다 호스트 컴퓨터의 데이터 전송속도가 빠르기 때문에 호스트 컴퓨터에서 보낸 데이터를 프린터에 일시 저장할 수 있는 메모리 공간을 말하며 프린터가 진행되는 동안 컴퓨터에서 다른 작업을 할 수 있도록 하여 데이터 전송시간을 절약하게 한다. 최근 들어 버퍼용량이 커지고 있는 추세이다.

바텀피드(Bottom Feed)는 연속용지를 위로 끌어당기는 역할을 하는 기능으로 프린터에는 이 기능을 위한 슬롯이 있는 것이 보통이며, 복잡한 양식의 무거운 종이를 처리해야 할 때 트랙터에 의해 종이 걸림을 막아준다.

DPI(Dot Per Inch)는 1인치 내에 인자할 수 있는 도트수를 말한다. 이것은 그래픽 해상도와도 밀접한 관계를 갖는다. 따라서 DPI가 클수록 그 밀도와 함께 프린트 상태는 좋아지지만 (해상도가 높아진다는 의미도 된다.) 속도가 늦어지기 때문에 이점도 처리업무에 관계한다고 볼 수 있다. 보통 LQ문자는 180 DPI, 드래프트문자는 60 DPI이며 한글, 한자에서는 180 DPI로 LQ문자와 같다. 레이저프린터는 300 DPI이고 그 이상의 것도 있다.

그리고 만일 한 라인에 DPI가 다른 문자들을 프린트할 경우, 즉 LQ문자와 드래프트문자가 한 라인에 함께 찍힐 때 프린트 헤드가 여러번 움직이는 것을 멀티패스(Multi Pass)라고 하는데 이러한 기능도 살펴보아야 한다.

절선이동(Skip of Perforation)

은 연속된 데이터를 프린트할 때 용지 절선부분에 프린트되는 것을 방지한다. 이때는 절선이동을 설정하여 프린트하면 절선을 기준으로 1인치 건너뛰어 프린트하게 된다. 또한 이 기능은 프린터에 따라 DIP SW로도 설정할 수 있다.

탭(Tap)은 프린트의 시작위치를 정하는 것으로 수직, 수평탭이 있다. 실제로 이 탭을 실행하지 않고도 그 시작위치를 조정할 수 있으나 그때마다 카운트하여 수직으로 이동할 때는 LF를, 수평일 때는 스페이스를 입력해야 하므로 프로그램이 복잡해지고 전송하는 데이터량도 많아지는 불편함이 있다. 따라서 이러한 수직, 수평탭의 기능이 프로그램을 간단히 하면서 속도도 빨라지게 한다.

그리고 고속의 프린터라고 하더라도 데이터 전송속도가 느린 컴퓨터에 연결되어 있다면 그 역할을 수행할 수 없다는 것을 유의하여 선택해야 한다.

●어떠한 기능을 가지고 있는가

그 다음에는 기능적인 면에서는 원하는 업무에 출력을 줄 수 있는지, 필요한 그래픽 출력에 필요한 그래픽 밀도를 어느정도 지원하는지, 글자의 품질은 사식 인쇄체와 유사하게 인쇄할 수 있는 글씨체인 NLQ와 다양한 폰트타입을 제공하는지 살펴야 한다. 특히 각종 정형화된 문서를 다루거나 워드프로세싱을 전문으로 다루는 곳에서는 글자체나 양식처리 기능이 다양한 프린터가 필요할 것이다.

이와 관련되어 프린터 헤드핀수를 구별하여 말하고 있는데 대체로 9핀이나 24핀으로 구분하고 있다. 이것은 핀사이의 간격에 따라 지칭하는 것이며 글씨가 인자될 때 텍스트모드에서는 9핀과 24핀이 별차이 없지만 그래픽모드에서는 핀사이의 간격이 해상도를 좌우하기 때문에 가격 결정의 중요한 요소가 되고 있다. 9핀 프린터는 핀간격이 1/72인치이고, 24핀 프린터는 1/180인치이다.

●어떤 문자가 출력되어 나오는가

문자의 측면에서는 다운로드 기능을 점검해야 하는데 이것은 실제 프린터 내부에 없는 문자를 정의하여 내장문자처럼 프린트할 수 있는 기능이다. 예를 들어 사용자가 'A' 코드를 그리스문자 'α'로 인자하기를 원한다면 정해진 매트릭스 크기에 따라 'α'의 이미지를 그려 데이터화시켜 프린터로 전송한다. 데이터는 램에 저장되어 있으며 'A'라는 코드의 입력을 받으면 'A'대신 'α'이미지를 프린트하게 된다. 대체로 프린터에 최대한 다운로드 할 수 있는 문자수는 127자 이다.

비트 이미지 기능은 보통의 프린터가 문자를 프린트하는 방식이 해당 코드를 입력받아 이 코드가 어떤 문자로 형상화 되어야 하는지 판별한 후에 롬에 저장되어 있는 폰트를 프린터 핀에 할당하여 문자를 프린트하는데 비하여 내장 폰트를 사용하지 않고 사용자가 핀에 도트 개념으로 데이터가 할당되도록 전송하여 프린트하는 방식이다. 실제 프린터가 그림을 인자하는 경우, 이것은 프린터에 내장되어 있는 데이터가 아니라 바로 비트이미지를 이용하는 것인데, 이것이 도트매트릭스 프린터의 큰 장점이다.

폰트는 문자 모양을 만드는 도트의 집합으로 구성되어 프린터의 롬에 내장된 문자패턴으로 컴퓨터에서 어느 문자코드를 보내면 이 코드를 해석하여 내장된 문자패턴을 읽어들여 인자를 실행한다. 그러므로 폰트가 다양할 경우 글씨체가 여러개 있기 때문에 다양한 인자체를 제공한다.

보통 한글에서는 고딕체와 명조체, 한자에서는 명조체, 영문에서는 드래프트, NLQ, 로만, 산세리프체 등이 제공되고 있다.

인자의 품질은 폰트 헤드의 왕복횟수와 헤드핀수를 증가시켜서 높일 수가 있는데 문자매트릭스(Character Matrix)는 NLQ 모드에서 한 문자를 표현하기 위해 소요한 격자의 크기로 표현된다. 문자 매트릭스를 구성하는 격자수가 클수록 그 문자 모양은 정밀해진다.

●호환성은 어떻게 해결하는가

이제까지의 사항들을 살펴서 어느정도 자신의 업무에 맞는 프린터를 구입할 수는 있지만 여기서 꼭 주의해야 할 것은 프린터와 소프트웨어의 호환성 문제이다. 프린터를 유용하게 이용하려면 그에 따른 소프트웨어가 필요한데 같은 소프트웨어라고 하더라도 기종에 따라 인쇄형태가 달라질 수 있으므로 소프트웨어의 호환성에 관하여 체크해 보아야 할 것이다.

현재 국내에 널리 보급되어 있는 도트매트릭스방식의 프린터는 대개 엡슨 기종이기 때문에 이 기종에 맞도록 대부분의 소프트웨어가 제공되고 있다. 만약 소프트웨어의 호환성에 문제가 있다면 그래픽이나 도표의 출력이라든가 글자의 모양이 다르게 변환되어 자신이 원하는 형태의 출력을 얻을 수 없다. 그러므로 구입할 프린터가 어느정도 자신의 목적에 맞는 소프트웨어와 호환성을 유지하는지 살펴야 할 것이다.

그리고 사용하고 있는 문자가 어떤 것인가도 살펴야 하는데, 한글에서는 조합형과 완성형과 KS한자의 지원유무, 그래픽문자는 어떤 것이, 영문은 어떠한 것이 지원되고 있는지 알아 보아야 한다.

호환성의 문제에서 부가적으로 고려할 사항은 인터페이스 문제이다. 프린터의 인터페이스 방식은 병렬방식과 직렬방식으로 두가지가 있다.

병렬방식은 8비트의 그룹을 1바이트의 데이터로서 전송하는데 각각의 데이터라인을 전기적인 로우, 하이상태로 프린터에 전송한다. 이 병렬방식은 프린터가 컴퓨터에 의해 조절할 수 있는 회선이 준비되므로 사용자 입장에서는 간단하다. 이 방식에서는 주로 센트로닉스 방식이 쓰이고 있다.

이에 비해 직렬방식은 한 데이터라인으로 일련의 데이터 스트림을 1바이트씩 인식해내는 방식으로 대체로 RS-232C 방식이 제공되고 있다.

프린터를 컴퓨터에 연결할 때는 어떤방식을 사용할 것인가를 결정해야 한다. 그리고 프린터의 인터페이스방식은 컴퓨터의 인터페이스 방법과도 연관을 가지므로 컴퓨터의 인터페이스 방식도 알아야 한다.

●그밖에 고려할 사항

그밖에 고려할 사항으로 용지에 대한 기능을 살펴 보아야 한다. 최대 복사매수는 원본의 2~5매 정도 인자를 허용하는데 카본용지를 사용해야 하는 불편함이 있다.

용지처리에 대한 기능에는 바텀피드, 라인피드, 폼피드가 있는데 라인피드는 캐리지 리턴과 비슷하며, 폼피드는 페이지 단위로 용지가 넘어가는 것이며 연속용지를 사용한다면 페이지 선두 부분으로 헤드를 옮겨 놓는 동작을 한다.

사용되는 용지는 연속용지와

낱장용지가 있다. 연속용지는 주로 한 페이지 길이가 11인치인데 프린터의 초기 설정치도 11인치로 되어 있다. 한 페이지는 폼피드 코드를 보냈을 때 용지가 다음 페이지까지 이동되는 양이다.

낱장용지를 사용하고자할 경우에는 낱장공급장치(CSF : Cut Sheet Feeder)를 설치하여 사용할 수 있다. 낱장용지장치는 주로 옵션으로 공급되고 있다.

그리고 리본이나 용지 등 기타 소모품 등을 구하는데 용이한지 알아보아야 한다. 특히 프린터 사용자는 누구나를 막론하고 리본을 갈아 끼워야 하기 때문에 이러한 사항에 대해서도 유의해야 한다. 리본은 비닐 수지에 카본을 코팅한 것과 나일론 천에 흑색 잉크류를 입힌 것이 있는데 이것은 사용하는 횟수가 증가함에 따라 글씨의 선명도가 약해진다. 리본의 길이는 길수록 사용할 수 있는 시간이 길지만 좁은 리본팩에 긴 리본을 끼울 경우는 출력 도중에 리본이 꼬일 수도 있으므로 적당한 길이의 리본선택에 유의해야 한다.

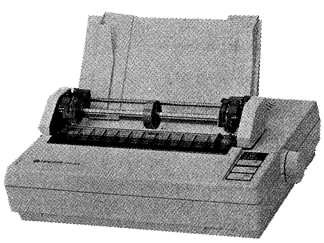
A/S는 어떻게 이루어지고 있나

어떤 제품을 선택하느냐에 따라 가장 큰 관심거리는 공급하고 있는 회사의 A / S 라고 할 수 있다. A / S는 프린터 판매처에서 담당하고 있으나 판매처의 직원들도 완벽한 유지보수 능력이 없기 때문에 문제가 생기게 되면 우선 판매처에 연락을 취하도록 하고 다시 판매처에서 제품을 생산하고 있는 업체에 직접 문의하는 방식을 택하고 있다. 또한 국내에서 생산되고 있지 않은 부품에 대하여는 외국에 신청하고 나서 그 부품이 국내에 도착하는 기간이 있기 때문에 시일이 걸린다. 그러므로 A / S에 대한 일반적인 절차와 그 기간에 관하여 알아보아야 할 것이다.

이제는 비용면을 살펴 보아야 할 것이다. 비용에 있어서는 가격대 성능비를 고려해야 할 것이다. 전체적인 유지보수 비용이 어느정도 소비되는지 계산하여 준비된 비용이 어느정도 되는지 파악하고 적당한 가격을 결정해야 한다.

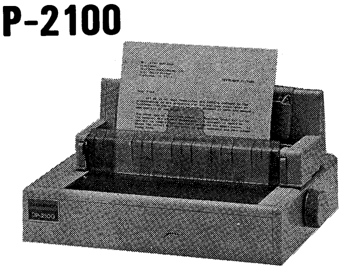
도트 매트릭스 프린터(9핀)

CPP-1500



- 공급처 : 대우전자
  (02) 754—0707
- 프린트방향 텍스트모드 : 양방향
  그래픽모드 : 단방향
- 프린트속도 한글 : 56CPS
  영문 : 150CPS
- 문자구성 : 9×9
- 행당 인자수 : 영문 80자 / 한글 60자
- 복사매수 : 원본+2
- 인터페이스 : 패러럴
- 가격 : 43만원(VAT 포함)

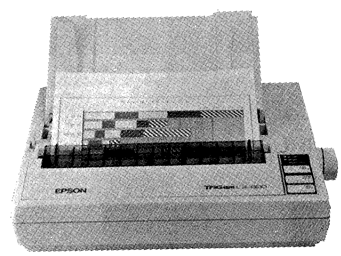
DP-2100



- 공급처 : 대우통신
  (02) 771-35
- 프린트방향 : 양방향(최단거리 이동)
- 프린트속도(10 CPI 기준)
  영문, 숫자 : 150 CPS(DRAFT), 50 CPS (NLQ)
  한글, 한자 : 25 CPS
- 문자체 영문 : DRAFT, ROMAN
  한글, 한자 : 고딕체
- 문자구성 영문, 숫자 : 9×9(DRAFT), 18×9(NLQ)
  한글, 한자 : 16×16, 14×16
- 사용문자 : 조합형 한글, 신문방송협회한자 5,170자, KS표준한자 4,888자, 영문, 숫자, 그래픽문자
- 행당 인자수 영문, 숫자 : 80자
  한글, 한자 : 53, 68자
- 복사매수 : 원본포함 2매
- 인터페이스 : 패러럴
- 가격 : 40만원(VAT 별도)

LX-800

- 공급처 : 삼보컴퓨터
  (02) 390—2114
- 프린트방향 텍스트모드 : 양방향
  이미지모드 : 단방향
- 프린트속도 한글 : 18.75 CPS



  영문 : 180 CPS
- 문자구성 한글 : 16×16
  영문 : 9×9
- 행당인자수 한글 : 60자
  영문 : 80자
- 인터페이스 : 패러럴
- 가격 : 39만원(VAT 별도)

FX-800

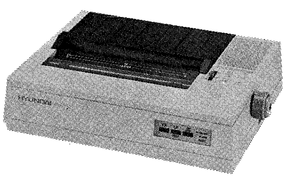
- 공급처 : 삼보컴퓨터
  (02) 390—2114
- 프린트방향 텍스트모드 : 양방향
  그래픽모드 : 단방향
- 프린트속도 한글 : 25 CPS(2 Pass)
  영문 : 240 CPS(Draft, Elite)
- 문자구성 한글 : 16×16
  영문 : 9×9
- 행당 인자수 한글 : 60자
  영문 : 80자
- 인터페이스 : 패러럴

• 가격 : 77만원(VAT 별도)

FX-1000

• 공급처 : 삼보컴퓨터
(02) 390－2114
• 프린트방향 텍스트모드 : 양방향
그래픽모드 : 단방향
• 프린트속도 한글 : 25 CPS(2 Pass)
영문 : 240 CPS(Draft, Elite)
• 문자구성 한글 : 16×16
영문 : 9×9
• 행당 인자수 한글 : 102자
영문 : 136자
• 인터페이스 : 패러렐
• 가격 : 100만원(VAT 별도)

HDP-910K



• 공급처 : 현대전자
(02) 741－1311
• 프린트방향 텍스트모드 : 양방향
그래픽모드 : 단방향
• 프린트속도 영문 : 180 CPS
한글 한자 : 46 CPS
• 문자구성 영문 : 11×9
한글 : 18×18
• 행당 인자수 : 80자
• 인터페이스 : 패러렐
• 가격 : 30만원(VAT 별도)

HDP-920K

• 공급처 : 현대전자
(02) 741－1311
• 프린트방향 텍스트모드 : 양방향
그래픽모드 : 단방향
• 프린트속도 영문모드 : 180 CPS
한글 한자 : 46 CPS
• 문자구성 영문 : 11×9
한글 : 18×18
• 행당 인자수 : 156자
• 인터페이스 : 패러렐
• 가격 : 40만원(VAT 별도)

도트 매트릭스 프린터(18핀)

HDP-1810K

• 공급처 : 현대전자
(02) 741－1311
• 프린트방향 텍스트모드 : 양방향
그래픽모드 : 단방향
• 프린트속도 영문모드 : 200 CPS
한글, 한자 : 134 CPS
• 문자구성 영문 : 12×18
한글 : 36×18
• 행당 인자수 : 80자
• 인터페이스 : 패러렐
• 가격 : 53만원(VAT 별도)

HDP-1820K

• 공급처 : 현대전자
(02) 741－1311
• 프린트방향 텍스트모드 : 양방향
그래픽모드 : 단방향
• 프린터속도 영문모드 : 200 CPS
한글, 한자 : 134 CPS
• 문자구성 영문 : 12×18
한글 : 36×18
• 행당 인자수 : 136자
• 인터페이스 : 패러렐
• 가격 : 60만원(VAT 별도)

도트 매트릭스 프린터(24핀)

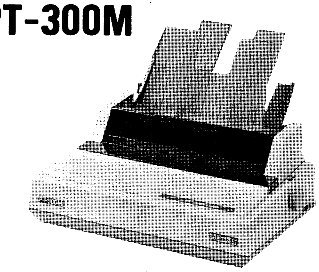
PRT-4550

• 공급처 : 금성사
(02) 701－1114
• 프린트방향 : 양방향 최단거리제어
• 프린트속도 한글 : 70 CPS
영문 : 185 CPS
• 문자구성 한글 : 24×24
영문 : 12×24
• 행당 인자수 한글 : 90자
영문 : 136자
• 복사매수 : 원본+4
• 인터페이스 : 패러렐, 시리얼
• 가격 : 160만원(VAT 별도)

PRT-5850

• 공급처 : 금성사
(02) 701－1114
• 프린트방향 : 양방향 최단거리제어
• 프린트속도 한글 : 200 CPS
영문 : 300 CPS
고속 : 450 CPS
• 문자구성 한글 : 24×24
영문 : 12×24
• 복사매수 : 원본+4
• 행당 인자수 한글 : 90자
영문 : 181자
• 인터페이스 : 시리얼, 패러렐
• 가격 : 270만원(VAT 별도)

PT-300M



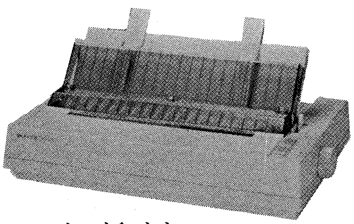
• 공급처 : 금성통신
(02) 787－4121
• 프린트방향 : 양방향 최단거리제어
• 프린트속도 한글 : 200 CPS
영문 : 300 CPS
• 문자구성 한글 : 24×24
영문 : 12×24
• 행당 인자수 영문 : 136자
한글 : 90자
• 복사매수 : 원본+4매
• 인터페이스 : 패러렐
• 가격 : 210만원(VAT 별도)

CPP-1600

• 공급처 : 대우전자
(02) 759－3814
• 프린트방향 : 양방향
• 프린트속도 한글 : 110 CPS
영문 : 220 CPS
• 문자구성 한글 : 24×24
영문 : 9×23

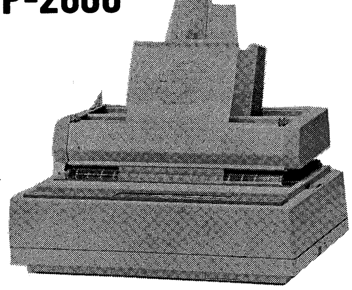
• 행당 인자수 한글 : 52자
영문 : 80자
• 문자종류 한글 : MSX-N바이트, 완성형, 조합형
한자 : 4888자
영자 : ASCII
• 복사매수 : 원본+2매
• 인터페이스 : 패러럴
• 가격 : 121만원(VAT 별도)

CPP-2500



• 공급처 : 대우전자
(02) 754—0707
• 프린트방향 : 양방향
• 프린트속도 한글 : 110 CPS
영문 : 220 CPS
• 문자구성 한글 : 24×24
영문 : 9×23
• 행당 최대인자수 한글 : 90자
영문 : 136자
• 문자종류 한글 : MSX-N바이트, 완성형, 조합형
한자 : 4888자
영자 : ASCII
• 복사매수 : 원본+2매
• 인터페이스 : 패러럴
• 가격 : 160만원(VAT 포함)

DP-2600



• 공급처 : 대우통신
(02) 771-35
• 프린트방향 : 양방향(최단거리 이동)
• 인쇄속도(10 CPI 기준)
영문, 숫자 : 330 CPS(HIGH SPEED DRAFT), 250 CPS(DRAFT), 100 CPS (LQ)
한글, 한자 : 80 CPS(고품질), 160 CPS (배속모드)
• 문자체 영문 : DRAFT, ITALIC, ROMAN
한글 : 명조체, 고딕체
한자 : 명조체
• 문자구성 영문, 숫자 : 12×8
한글, 한자 : 24×26
반각문자 : 24×13
• 사용문자 : 조합형 / KS완성형 한글 신문방송협회(5,170자) / KS표준(4,888) 한자
• 행당 인자수 영문, 숫자 : 132자
한글, 한자 : 90자
• 복사매수 : 원본포함 5매
• 인터페이스 : 패러럴, 시리얼
• 가격 : 170만원(VAT 별도)

SP-2000G

• 공급처 : 동양시스템
(02) 587-2541
• 프린트방향 : 양방향 최단거리 인자
• 프린트속도 한글 : 120 CPS
한자 : 120 CPS
영문 : 200 CPS
• 문자구성 한글 : 24×24
영문 : 13×18
• 행당 인자수 한글 : 52자
영문 : 80자
• 사용문자 영문, 숫자 : 96 ASCII
한글 : 7비트 과기처 표준, 완성형, 조합형
한자 : 국가표준형
• 인터페이스 : 패러럴
• 가격 : 180만원(VAT 별도)

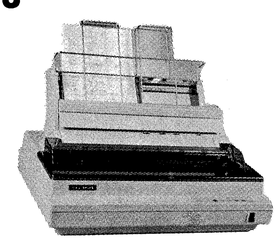
9250

• 공급처 : 동양시스템
(02) 578-2541
• 프린트방향 : 양방향 최단 제어
• 프린트속도 한글 : 130 CPS
한자 : 80 CPS
영문 : 250 CPS
• 문자구성 한글 : 18×24
한자 : 24×24
영문 : 12×24, 18×24
• 행당 인자수 한글 : 136자
한자 : 102자
영문 : 136자
• 복사매수 : 원본+4매
• 인터페이스 : 패러럴
• 가격 : 182만원(VAT 별도)

SP-2000

• 공급처 : 동양시스템
(02) 578-2541
• 프린트방향 : 양방향 최단거리
• 프린트속도 한글 : 120 CPS
한자 : 120 CPS
영문 : 200 CPS
• 문자구성 한글 : 24×24
영문 : 13×18
• 행당 인자수 한글, 한자 : 90자
영문 : 136자
• 사용문자 영문, 숫자 : 96 ASCII
한글 : 조합가능한 모든한글
한자 : 1692자
• 복사매수 : 원본+4매
• 인터페이스 : 패러럴
• 가격 : 180만원(VAT 별도)

P450



• 공급처 : 동양시스템
(02) 587-2541
• 프린트방향 : 양방향 최단거리 제어방식
• 프린트속도 고속한글, 한자 : 220자
영문, 숫자 : 고속321자
영문, 숫자(반각) : 고속450자
• 문자구성 한글, 한자 : 24×24
영문, 숫자 : 16×24
영문, 숫자 : 12×24
• 행당 인자수 한글 : 90자
영문, 숫자 : 136자
• 복사매수 : 원본+4매
• 인터페이스 : 패러럴

• 가격 : 280만원(VAT 별도)

LQ-850

• 공급처 : 삼보컴퓨터
(02) 390-2114
• 프린트방향 : 양방향
• 프린트속도 영문 : 264 CPS
한글 : 110 CPS
• 문자구성 한글 : 24×24
영문 : 29×23
• 행당 인자수 영문 : 80자
한글 : 52자
• 복사매수 : 원본+3
• 인터페이스 : 패러럴
• 가격 : 110만원(VAT 별도)

LQ-1550

• 공급처 : 삼보컴퓨터
(02) 390-2114
• 프린트방향 : 양방향
• 프린트속도 영문 : 264 CPS
한글 : 110 CPS
• 문자구성 영문 : 29×23
한글 : 24×24
• 행당 인자수 영문 : 136자
한글 : 90자
• 복사매수 : 원본+3매
• 인터페이스 : 시리얼, 패러럴
• 가격 : 140만원(VAT 별도)

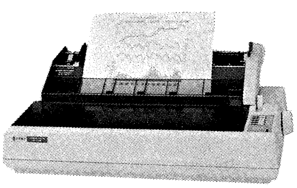
LQ-2550H

• 공급처 : 삼보컴퓨터
(02) 390-2114
• 프린트방향 : 양방향
• 프린트속도 영문 : 400 CPS
한글 : 148 CPS
• 문자구성 영문 : 9×23
한글 : 24×24
• 행당 인자수 영문 : 80자
한글 : 52자
• 리본카트리지 : 흑색, 칼라(흑색, 청색, 적색, 노랑)
• 인터페이스 : 시리얼, 패러럴
• 가격 : 195만원(VAT 별도)

SP-610

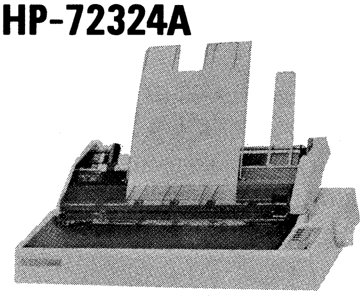
• 공급처 : 삼성전자
(02) 599-2643
• 프린트방향 : 양방향
• 프린트속도 한글, 한자 : 135 CPS
영문 : 324 CPS
• 문자구성 한글, 한자 : 24×23
영문 : 9×23
• 행당 인자수 한글, 한자 : 102자
영문 : 136자
• 사용문자 한글 : 조합가능한 글자
한자 : 5170자
영문 : 96 ASCII
• 복사매수 : 원본+3매
• 인터페이스 : 패러럴, 시리얼
• 가격 : 450만원(VAT 별도)

SP-570B



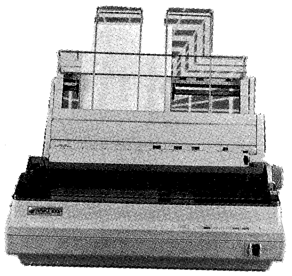
• 공급처 : 삼성전자
(02) 559—2643
• 프린트방향 : 양방향
• 프린트속도 한글 : 84 CPS
영문 : 222 CPS
• 행당 인자수 : 136자
• 사용문자체 한글 : 명조체
영문 : 드래프트, 고딕, 이태릭체등
• 복사매수 : 원본+3매
• 인터페이스 : 패러럴, 시리얼
• 가격 : 142만원(VAT 별도)

HP-72324A



• 공급처 : 삼성휴렛팩커드
(02) 784-2666
• 프린트방향 : 양방향
• 프린트속도 한글 : 84 CPS
영자 : 220 CPS
• 사용문자 영자 : ASCII
한글 : 2350자
한자 : 4888자
• 복사매수 : 원본+3매
• 인터페이스 : 시리얼, 패러럴
• 가격 : 158만5천4백원(VAT 별도)

HART-300P



• 공급처 : (주) 정원시스템
(02) 542-3675
• 프린트방향 : 양방향 최단제어
• 프린트속도 한글 : 150 CPS
영문 : 300 CPS
• 문자구성 한글 : 24×24
영문 : 24×12
• 행당 인자수 한글 : 90자
영문 : 136자
• 사용문자 : 영문반각, 보통영문, 전각한글
• 복사매수 : 원본+4매
• 인터페이스 : 패러럴
• 가격 : 180만원(VAT 별도)

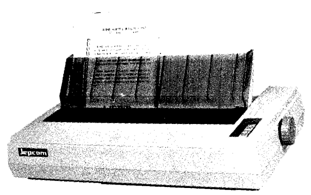
HART-428P

• 공급처 : (주) 정원시스템
(02) 542-3675
• 프린트방향 : 양방향
• 프린트속도 영문 : 428 CPS
한글 : 214 CPS
• 문자구성 영문 : 24×16
한글 : 24×24
• 행당 인자수 영문 : 136자
한글 : 90자
• 복사매수 : 원본+4매
• 인터페이스 : 패러럴
• 가격 : 280만원(VAT 별도)

UPT-300

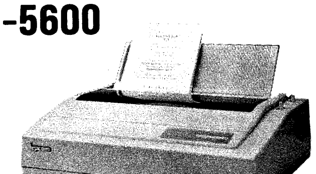
- 공급처 : 유니온시스템
  (02) 533-6213
- 프린트방향 : 양방향 최단거리인자
- 프린트속도 한글, 한자 : 220자
  영문, 숫자 : 300자
- 문자구성 한글 : 22×24
  영문 : 13×24
- 행당 인자수 : 한글, 한자 : 90자
  영문, 숫자 : 136자
- 사용문자 : 한글, 한자, 영문, 숫자
- 복사매수 : 원본+4매
- 인터페이스 : 패러렐
- 가격 : 280만원(VAT 별도)

JP-3400



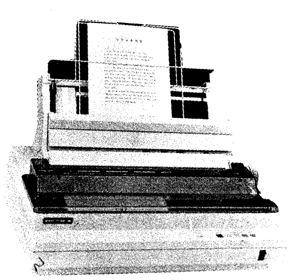
- 공급처 : 제일정밀공업(주)
  (02) 785-4161
- 프린트방향 : 양방향
- 프린트속도(12 CPI 기준)
  영문, 숫자 : 200 CPS
  한글, 한자 : 66 CPS
  한글, 한자배속 : 132 CPS
  한글, 한자축소문자 : 146 CPS
- 문자구성 영문, 숫자 : 13×19
  한글, 한자 : 24×24
  한글, 한자반각 : 12×24
  한글, 한자축소문자 : 7×19
- 행당 인자수 영문:136자
  한글, 한자:90자
- 복사매수 : 원본포함 4매
- 인터페이스 : 패러렐, 시리얼(선택)
- 가격 : 140만원(VAT 별도)

JP-5600



- 공급처 : 제일정밀공업(주)
  (02) 785-4161
- 프린트속도(12 CPI 기준)
  영문, 숫자 : 486 CPS
  한글, 한자 : 162 CPS
  한글, 한자배속 : 324 CPS
  한글축소문자 : 292 CPS
- 문자구성(세로×가로)
  영문, 숫자 : 19×13
  한글, 한자 : 24×24
  반각문자 : 24×12
  축소문자 : 19×7
- 행당 인자수 영문 : 136 자
  한글, 한자 : 90자
- 인자방향 : 양방향
- 복사매수 : 원본포함 5매
- 잉크리본 : 카트리지 타입(흑, 4색 칼라)
- 인터페이스 : 패러렐, 시리얼(선택)
- 가격 : 290만원(VAT 별도)

FPRT-200

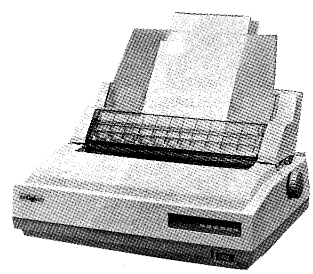


- 공급처 : FAST SYSTEM
  (02) 552-9569
- 헤드핀 수 : 24핀
- 문자구성(1 / 160인치 기준)
  영문, 숫자 : 24×16, 24×12, 24×24
  한글, 한자 : 24×24, 24×32
- 행당 인자수(1 / 160인치 기준)
  영문, 숫자 : 136자
  한글, 한자 : 90자
- 인자간격(1 / 160인치 기준)
  영문, 숫자 : 10 CPI, 13.3 CPI, 6.67 CPI
  한글, 한자 : 6.67 CPI, 5 CPI
- 문자종류 : 한글, 한자, 영문, 숫자, 부호, KSC5601의 모든 문자세트
- 문자체 : 명조체, 고딕체
- 인자방향 : 양방향(최단거리 이동)
- 인자속도 고속영문 : 227 CPS(10 CPI)
  고속한글, 한자 : 152 CPS(6.67 CPI)
  고밀도영문 : 114 CPS(10 CPI)
  한글, 한자 : 75 CPS(6.67 CPI)
- 복사매수 : 원본포함 5매
- 인터페이스 : 패러렐, 시리얼

QP3287D

- 공급처 : 큐닉스
  (02) 865-0511
- 프린트방향 : 양방향
- 프린트속도 영문 : 150 CPS
- 문자구성 한글 : 24×24
  영문 : 12×24
- 행당 인자수 한글 : 66자
  영문 : 132자
- 복사매수 : 원본+4매
- 인터페이스 : 패러렐

QP5550(IBM 5550용 프린터)



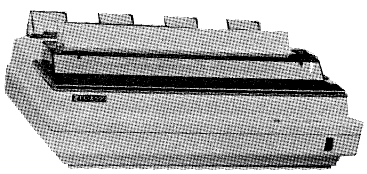
- 공급처 : 큐닉스
  (02) 865-0511
- 프린트방향 : 양방향
- 프린트속도 한글 : 70 CPS
  영문 : 140 CPS
- 문자구성 : 24×24
- 복사매수 : 원본+4매
- 인터페이스 : 패러렐

PC24

- 공급처 : 큐닉스
  (02) 865-0511
- 프린트방향 : 양방향, 최단거리인자
- 프린트속도 한글 : 87 CPS
  영문 : 260 CPS
- 문자구성 한글 : 24×24
  영문 : 11×17, 24×24
- 행당 인자수 한글 : 90자
  영문 : 136자

• 사용문자 한글 : 조합형, 완성형
한자 : 2바이트 5,170자
• 복사매수 : 원본+4매
• 인터페이스 : 패러렐

FC-K300



• 공급처 : 화인텍
(02) 699-5711
• 프린트방향 : 양방향 최단거리
• 프린트속도 한글, 한자 : 214 CPS
영문 : 321 CPS
• 문자구성 한글 : 24×22
영문 : 16×24
• 행당 인자수 한글 : 90자
영문 : 136자
• 복사매수 : 원본+4매
• 사용문자 : 한글, 한자, 영문, 숫자, 부호
• 인터페이스 : 패러렐
• 가격 : 260만원(VAT 별도)

FC-Q240

• 공급처 : 화인텍
(02) 699-5711
• 프린트방향 : 양방향
• 프린트속도 한글 : 152 CPS
영문 : 227 CPS
• 문자구성 한글 : 22×22
영문 : 24×13
• 행당 인자수 한글 : 90자
영자 : 136문자
• 사용문자 : 한글, 한자, 영문
• 복사매수 : 원본+4매
• 인터페이스 : 패러렐
• 가격 : 160만원(VAT 별도)

HDP-1625

• 공급처 : 현대전자
(02) 741-1311
• 프린트방향 : 양방향
• 프린트속도 영문 : 264 CPS
한글 : 110 CPS
• 문자구성 영문 : 29×23
한글 : 24×24
• 행당 인자수 영문 : 80자
한글 : 52자
• 복사매수 : 원본+3매
• 인터페이스 : 패러렐
• 가격 : 110만원(VAT 별도)

HDP-1645

• 공급처 : 현대전자
(02) 741-1311
• 프린트방향 : 양방향
• 프린트속도 영문 : 264 CPS
한글 : 110 CPS
• 문자구성 영문 : 29×23
한글 : 24×24
• 행당 인자수 영문 : 136자
한글 : 90자
• 복사매수 : 원본+3매
• 인터페이스 : 패러렐
• 가격 : 140만원(VAT 포함)

HDP-3400

• 공급처 : 현대전자
(02) 741-1311
• 프린트방향 : 양방향
• 프린트속도 영문 : 200 CPS
한글 : 66 CPS
• 문자구성 영문 : 13×19
한글 : 24×24
• 행당 인자수 영문 : 136자
한글 : 90자
• 복사매수 : 원본+3매
• 인터페이스 : 패러렐
• 가격 : 135만원(VAT 별도)

HDP-4500

• 공급처 : 현대전자
(02) 741-1311
• 프린트방향 : 양방향
• 프린트속도 영문 : 240 CPS
한글 : 80 CPS
• 행당 인자수 영문 : 136자
한글 : 90자
• 복사매수 : 원본+3매
• 인터페이스 : 패러렐
• 가격 : 170만원(VAT 별도)

프린터 사용시 발생하는 고장 및 응급처치

이제는 웬만한 사무실은 물론 가정에도 컴퓨터가 보편화되고 있는 실정이다. 따라서 이전에는 꼭 필요한 경우 이외에는 제대로 구입할 수 없었던 주변기기를 이제는 거의 꼭 구입해야 하는 필수품이 되고 있다. 그 중 대표적인 것으로 프린터를 꼽을 수 있다.

몇년전까지만 하더라도 가격이 상당히 비쌌기 때문에 구입하는데 많은 어려움이 있었다. 그러나 이제는 인자 기능의 문제만 따지지 않는다면 20만원대의 프린터도 구입할 수 있게 되었다. 그러나 컴퓨터의 국산화가 많이 진척되었다는 요란한 선전과는 달리 프린터는 아직 국산화율이 상대적으로 떨어지고 있다. 그 이유는 컴퓨터가 전자부품 위주라면 프린터는 정밀기계 위주이기 때문에 상당한 투자와 시간을 필요로 하는데 우리의 여건상 어려운 실정이기 때문이다. 그렇기 때문에 외국 부품을 들여다가 조립하면서 다시 한글에 맞게 고치고, 또 우리 전기 사정상 전압등이 고르게 공급되지 못한 실정이라 고장도 외국보다 더 잦은 것이 사실이다.

그러나 생산자나 수리센터 입장에서 본다면 프린터 설명서를 제대로 읽지 않고 사용하기 때문에 일어나는 고장도 상당히 많다고 한다. 예를 든다면 110V 전용이라고 쓰여진 것을 제대로 확인하지 않고 220V에 꽂아서 일어나는 사고가 한두 건이 아니다. 이렇게 사용자의 부주의로 해서 일어나는 고장은 대개 무상보증 기간이라도 수리비를 받게 마련이다. 이것만 보더라도 조금만 신경을 쓴다면 컴퓨터를 놀리는 시간과 쓸데없는 경비를

줄일 수 있을 것이다.

●프린터를 구입할 때

접속방식이 패러럴(병렬)방식인가 시리얼(직렬)방식인가 아니면 겸용인가를 확인한다.

처음 프린터 구입시에는 대개 현재 사용하는 컴퓨터에 맞는 방식으로 구입하지만 컴퓨터를 확장하다 보면 다른 접속방식이 필요하기 마련이다. 그런데 흔히 구입자들이 착각하기 쉬운 것이 무엇 무엇 겸용이라 하는 것이 대부분 선택사양이라는 점이다. 절대 오해하지 말고 구입전에 선택 사양인지 필수 사양인지를 꼭 확인하도록 한다. 어떤 기종은 병렬방식과 직렬방식을 같이 사용 못하는 전용기종인 경우도 있으니 주의를 바란다.

●사용 전압을 확인한 다음 구입토록 한다.

현재 상용되는 전압은 대개 110V이지만 간혹 220V인 경우도 있으니 현재 사용하고자 하는 곳의 전압을 확인한 다음 거기에 맞는 프린터를 구입하도록 한다.

●프린터가 켜지지 않는다.

프린터에는 여러가지 표시등이 있기 마련이다. 프린터를 사용하기 위해 켰더니 아무런 표시등이 켜지지 않거나 아무런 동작도 하지 않는다면 이것처럼 난감한 경우도 없을 것이다. 실은 절대 당황할 일이 아니다. 우선 전원코드를 확인한다. 프린터에서 가까운 쪽에 있는 전원부터 하나씩 확인한다.

이렇게 해서 전원에 이상이 없다면 대부분 프린터 내부에 있는 전원 휴즈가 끊어진 경우이다. 이럴 때는 프린터 케이스를 분해하고 휴즈를 교환한 다음 다시 케이스를 그대로 조립할 자신이 있다면 한번 해 볼만하다. 이때 주의할 점은 끊어진 휴즈와 똑같은 규격으로 바꿔야 한다는 점이다. 낮은 규격이라면 금방 끊어질 것이고 높은 규격이라면 다른 더 큰 고장이 일어날 수 있다. 휴즈를 교환한 다음 케이스를 덮지 말고 컴퓨터에서 내보내는 데이터를 제대로 찍는가를 확인한 다음 조립하도록 한다. 같은 규격의 휴즈로 교환했는데도 자꾸 끊어진다면 다른 곳에 이상이 있는 것이니 A/S도움을 요청한다.

●프린터 헤드가 움직이지 않는다.

프린터를 켜면 제일 먼저 헤드의 첫 위치를 찾아간다. 그런데 이 동작을 하지 않는다면 우선 리본을 제거하고 프린터를 작동시켜 본다. 이때 정상적으로 동작한다면 일단 프린터는 이상이 없다고 보아도 무방하다.

이제 살펴 볼 것은 리본과 리본케이스이다. 리본은 프린터 헤드가 종이에 글자를 찍으면서 돌아가는데 이때 리본케이스가 리본을 돌려주는 축과 적절히 물리지 않으면 헤드가 리본의 같은 부분만 때려 리본 헝겊부분이 풀어지게 되고 그 풀어진 실이 리본케이스 기어에 감겨서 헤드가 움직이지 않는 사고가 일어나게 되는 것이다. 그렇지 않으면 원래 그 프린터 리본케이스에 맞는 적절한 길이의 리본을 사용해야 하는데 너무 긴 리본을 사용했다면 이것 또한 프린터 헤드를 멈추게 하는 원인이 된다. 리본케이스를 분해해 보면 한쪽으로 몰려있는 것을 확인할 수 있을 것이다. 그러므로 해당 프린터에 맞는 적절한 길이의 리본을 사용하도록 하자.

또 리본을 오래 사용하면 글자가 흐려지는데 이때 만년필등의 잉크를 리본에 뿌려 사용하지 말자. 잉크 찌꺼기가 프린터 헤드에 끼어 헤드핀 수명을 단축시킬 수 있다. 그리고 리본은 될 수 있는한 프린터 제조처에서 권하는 제품을 사용하도록 한다.

리본을 확인했는데도 이상이 있다면 컴퓨터와 연결된 데이터선을 확인한다.

●프린터를 켜보니 ON LINE램프가 켜지지 않는다.

대부분 프린터 고장이라고 생각되지만 컴퓨터나 케이블에 이상이 있는 경우도 많기 때문에 여러가지 응급처치를 해보면서 어떤 요소가 고장인지를 확인한다.

우선 프린터와 컴퓨터의 전원을 끈다. 그리고 서로 연결된 데이터선을 뺀다음 프린터 스위치를 ON시켜서 동작시켜 본다. 이때 프린터 헤드가 정상적으로 동작한다면 (흔히 정상일 때 움직이는 것처럼 몇번 움직이는 것, 이것은 헤드가 처음에 찍어야 할 자리를 찾기 위한 초기동작이다.) 프린터는 일단 이상이 없으리라 판단할 수 있다. 여기서 프린터의 셀프테스트(자기진단)를 해보면 더욱 확실한 진단방법이 된다.

그러나 그래도 의심스럽다면 프린터를 다른 컴퓨터에 연결해 보면서 시험해 보면 될 것이다.

다른 경우로써 컴퓨터와 연결된 데이터선이 빠진 경우도 있으니 확인한다. 그리고 처음 프린터를 켰을 때에는 제대로 동작하나 컴퓨터에서 데이터를 보낼 때 ON LINE만 켜지면서 실제 아무것도 찍히지 않는다면 대부분 프린터 고장이니 대리점이나 프린터제조처에 증상 등을 말하고 A/S를 요청한다.

●프린터를 사용할 때

프린터 손잡이 위치를 주의하여야 한다. 프린터용지를 돌려주는 손잡이가 의외로 고장이 많아 교환을 요구하는 경우가 많다. 이 손잡이가 파손되어 있다는 것은 프린터를 상당히 거칠게 사용하고 있는 것으로 볼 수 있지만 더 큰 이유는 프린터 설치 장소에 있다고 본다. 프린터 손잡이가 튀어 나와 있기 때문에 어떤 물체가 지나갈 때 이 손잡이에 충격을 주게 되어 파손되는 경우가 대부분이어서 프린터를 안전한 방향으로 돌려서 사용하도록 하자.

●프린터 속도가 느리지만 다른 작업을 하기 위해 프린터를 계속 찍히게 할 수 있는 방법이 있다.

요즈음 컴퓨터 관련 잡지를 보면 프린터 버퍼라는 명칭으로 광고하고 있는 제품이 많이 있을 것이다. 이것은 프린터가 찍을 데이터를 한꺼번에 많이 받아들여 그 내용을 다 찍을 때까지 컴퓨터는 다른 작업을 계속할 수 있게 만든 것이다. 많이 이용해 볼 만 한 것이다.

이 프린터 버퍼는 외장용과 컴퓨터 내장용 2가지가 있는데 외장용은 컴퓨터 내부에 빈 공간이 없을 때 편하고 내장용은 외부 전원코드나 부피등에 있어서 지저분하게 따로 돌아다니지 않기 때문에 편하다.

프린터 버퍼 용량은 보통 64K에서 주문에 따라 1M까지 생산하고 있다. 그러나 64K 이상이면 사용상 하등의 문제가 없으리라 생각된다. 필자가 사용하고 있는 것은 내장형(64K Byte)인데 프로그램을 개발하면서 사용할 때 아주 편리하다.

참고로 데이터 스위치라는 장치도 있다. 이것은 프린터 한대에 컴퓨터 여러대를 부착하여 사용할 수 있게 해주는 장치이다. 동시에 찍을 수 있게 해주는

것이 아니고 사용자가 필요한 쪽으로 스위치를 돌려서 컴퓨터와 프린터를 1대 1로 교환하면서 사용할 수 있게 한 것이다.

●프린터 사용중 컴퓨터나 프린터가 죽어버리는 경우가 있다.

이때 사용하는 프로그램이 베이직이라면 베이직에서 일정한 시간을 두고 프린터 제어를 하기 때문에 발생하는 문제인 것이다.

베이직 프로그램의 첫 부분에

```
DEF SEG=&H40:POKE&H78,
&H78, 0:DEF SEG
```

를 삽입하면 거의 해결된다. 그런데 요즘 컴퓨터의 CPU속도를 높인 기종이 많아져서 위 방법을 사용해도 멈추는 경우가 있을 것이다. 이때는 베이직을 사용하기 전에 그 컴퓨터의 스피드를 느리게 할 필요가 있다. 컴퓨터에 있는 스위치로 속도를 줄이든가 프로그램을 통하여 10MHz 시스템이라면 8MHz나 5MHz로 낮추어 사용하면 될 것이다. 만약 다른 언어에서 사용할 때 프린터나 컴퓨터가 멈춘다면 그 언어의 매뉴얼을 참조하는 수밖에 없다.

●프린터 헤드가 종이의 두께에 따라 찍을 수 있게 잘 조절하여 사용한다.

프린터를 사용하다 보면 거의 대부분 한 장으로 된 용지만 사용하기 때문에 어쩌면 헤드 밀착도를 조절하는 레버를 건드리지 않을지는 몰라도 간혹 사용자가 복사용지를 사용하고 나서 원위치로 풀지 않고 사용하다 보면 프린트하는 속도가 떨어지거나 갑자기 소음이 종전보다 상당히 커져 있는 경우가 있다. 이런 현상이 계속되다 보면 프린터 핀이 빨리 소모되므로 프린터 헤드 높이를 조절하는 레버를 종전의 상태로 조정해야 한다.

●프린트 용지를 뺄 때에는 절대 뒤로 돌려서 빼지 말라.

프린터를 잘 살펴보면 헤드 밑면에 밑에서 올라오는 종이가 로울러에 밀착되도록 하게 하는 얇은 강철판을 볼 수 있다. 종이를 자주 뒤로 돌리다 보면 이 강철판이 안으로 말리거나 파손되는 등 편편하지 않게 되어 종이가 자주 찢어지거나 정상적으로 올라오지 않게 된다. 그리고 될 수 있는대로 두꺼운 용지는 피하는 것이 좋다.

●낱장용지를 사용할 때 올라오는 것이 똑바르지 않다.

흔히 프린터는 연속용지만 사용하는 것으로 알고 있는데 낱장용지도 사용할 수 있다. 이때는 낱장용지를 밀어주는 로울러가 있기 마련이다. 이 로울러가 2개 이상 있는 것이 보통인데 프린터 제조상 이 로울러가 정확하게 같은 힘으로 밀어 주게 되지만 상당히 어려운 것이다. 그러나 대개 비슷한 힘으로 밀어주기 때문에 낱장 용지를 사용하더라도 거의 똑바르게 올라오지만 심하게 비틀려 올라온다면 이 부분에 대하여 A/S요청을 해야 한다.

●화면에는 한글이 나오는데 프린터로 찍을 때는 한글이 찍히지 않는다.

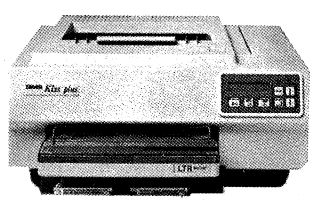
화면에 한글이 나오더라도 프린터는 이 글자가 영문인지 한글인지 전혀 알 수 없다. 어떤것으로 찍더라도 한글을 찍을려면 프린터 한글 모듈을 한글 상태로 놓거나 프로그램에서 한글 인식코드를 프린터로 보내야 한다.

위 2 가지 방법 중 아무거나 사용하더라도 한글을 찍는데는 상관이 없으나 어떤 특정한 유틸리티나 그래픽을 사용할 수 없는 경우가 많으니 이것을 많이 사용하는 사용자라면 프로그램에서 한글인식코드를 프린터로 보내야 한다.

□자료협조 :
엄수호 / 삼보컴퓨터 기술부

레이저 프린터

QMS KISS PLUS



- 공급처 : 대우통신 (02) 771-35
- 엔진 : CANON SX
- 프린트방식 : 반도체 레이저에 의한 건식 전자사진방식
- 해상도 : 300×300 DPI(Dot Per Inch)
- 프린속도 : 8PPM(Page Per Minute)
- 듀티사이클 : 5,000페이지 / 월
- 토너 카트리지 수명 : 4,000페이지 / 월
- 시동시간 : 60초
- 사용용지 : A4(210×297㎜), LETTER (216×279㎜), LEGAL(216×356㎜)
- 영문폰트 : 17
- 메모리 : DRAM-1MB / 2.5MB USER MEMORY-750KB / 2.1MB
- 프린트방향 : Portrait, Landscape
- 인터페이스 : 시리얼, 패러럴
- 가격 : 339만5천원(VAT 별도)

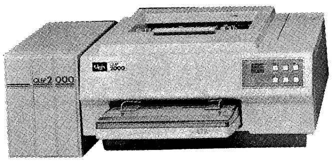
QMS-PS 810



- 공급처 : 대우통신 (02) 771-35
- 엔진 : CANON SX
- 프린트방식 : 반도체 레이저에 의한 건식 전자사진방식
- 해상도 : 300×300 DPT(Dot Per Inch)
- 프린트속도 : 8PPM
- 듀티사이클 : 5,000 페이지 / 월
- 토너 카트리지 수명 : 4,000페이지 / 월
- 시동시간 : 60초
- 사용용지 : A4, LETTER, LEGAL
- 영문 폰트 : 35
- 인자방향 : Portrait, Landscape
- 메모리 : 2MB RAM, 1MB ROM
- 인터페이스 : RS-232C / RS422 / Apple Talk / 센트로닉스규격

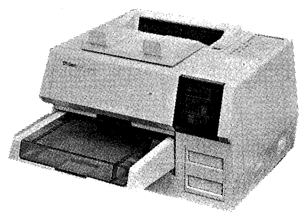
• 가격 : 622만4천원(VAT 별도)

QLBP2000



• 공급처 : 큐닉스
(02) 783-4301
• 형식 : 탁상형 페이지프린터
• 프린트방식 : 건식전자 사진방식
• 프린트속도 : 분당 최대8매
• 해상도 : 300 DPI
• 사용문자 한글 : (명조, 고딕) 모든한글
한자 : (명조) 5,170자
영문 : ASCII
• 문자구성 : 40×40, 32×32
• 그래픽기능 : Full Graphic
• 칼라 : 단색(흑색, 암갈색) / 카세트 교환방식
• 인터페이스 : 시리얼

LBP-4080



• 공급처 : 삼보컴퓨터
(02) 390-2114
• CPU : 68000(10MHz)
• 메모리용량 : 1.5MB~2.0MB
• 문자체 한글 : 명조체, 고딕체
한자 : 명조체
• 문자크기 한글, 한자 : 40×40, 24×24
영문 : 40×20, 24×16
• 문자회전 : 90°, 180°, 270°
• 인터페이스 : 패러렐
• 가격 : 500만원(VAT 별도)

M6210

• 공급처 : 엘렉스컴퓨터
(02) 780-4545
• CPU : MC68000(12MHz)
• 메모리용량 RAM : 2MB
ROM : 1MB
• 분당 프린트속도 : 8장
• 해상도 : 300×300 DPI
• 인터페이스 : 애플데스크탑버스, 시리얼
• 가격 : 550만원(VAT 별도)

M6215

• 공급처 : 엘렉스컴퓨터
(02) 780-4545
• CPU : MC68000(16.67MHz)
• 메모리용량 RAM : 2MB
ROM : 1MB
• 분당프린트속도 : 8장
• 해상도 : 300×300 DPI
• 인터페이스 : 애플데스크탑버스, SCSI, RS-232C
• 가격 : 790만원(VAT 별도)

보내는 사람

□□□-□□□

월간 퍼스널컴퓨터 하이테크

서울특별시 용산구 갈월동 17

퍼스널컴퓨터 편집부앞

1 4 0 - 1 5 0

애독자페이지

관심있는 분야		게재를 원하는 기사	

□ 컴퓨터에 관한 알고 싶은 질문

□ 본지에 바라는 의견

정기구독신청서

구독신청방법	□ 은행지로 □ 우체국 대체구좌 □ 소액환 □ 기타		
구독기간	년 월호부터 년 월호까지	구독신청일자	

□ 받으실 분의 이름과 주소(우편번호)

☎ (—)

※ 이곳에 풀칠을 하여 뒷면 수신란이 나오게 접어서 우체통에 넣어 주세요(우표부착 불필요).

애독자 설문지

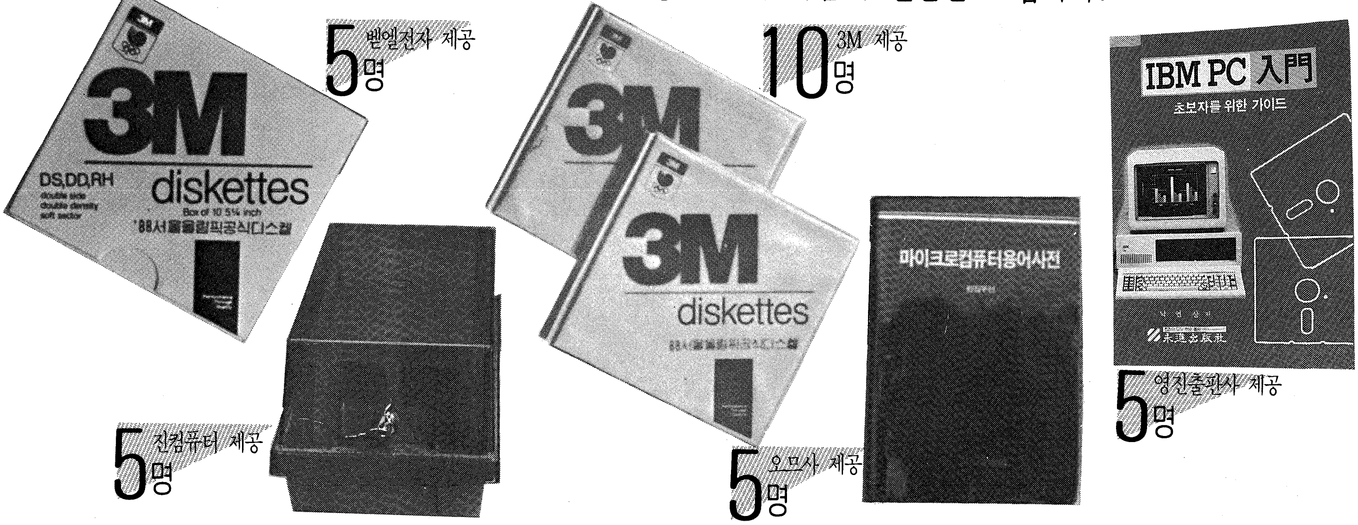
책의 편집에 반영하고자 하오니 성의껏 적어서 보내주시기 바랍니다.

구분	항목	내용
작성자	이름	나이 / 직장 / 성별
	주소	☎ (–)
사용기종	8비트	16비트 / 32비트
	주변기기	□ 모니터(컬러, 단색) □ 드라이브(HDD, FDD) □ 프린터(9핀, 24핀, 레이저) □ 인터페이스카드( )
	사용용도	□ 취미 □ 업무 □ 교육 □ 학습 □ 기타 / 사용경력
	사용언어	□ 베이직 □ C □ 파스칼 □ 기계어 □ 기타
	사용분야	사용 소프트웨어
책구입방법	구입방법	□ 정기구독 □ 서점구입(서점) □ 증정 □ 기타
	구입동기	□ 정기구독 □ 필요한 내용이 있어서 □ 주위에서 권해서 □ 기타
	본지외 구독잡지	□ 없다 □ 있다(잡지명 :) / 원하는 발행일자
	애로사항	□ 가격이 비싸다 □ 종이 질이 나쁘다 □ 인쇄상태가 나쁘다 □ 기타
	□ 본지 구독을 추천하고 싶은 사람이 있다면(이름과 주소)	☎ (–)
이번달에 좋았던 기사(2가지)	제목	제목
	□ 이유	□ 이유
이번달에 싫었던 기사(2가지)	제목	제목
	□ 이유	□ 이유

자르는 선

애독자 선물대잔치

월간 퍼스널컴퓨터는 독자 여러분의 의견을 소중히 하여 책의 편집에 반영하고자 합니다.
애독자 설문지를 사용하여 질의내용에 성의껏 답하시어
15일까지 보내주시면 30명을 뽑아 다음의 선물을 드립니다.



당첨자 명단

● 디스켓 보관함 ●

성상기 서울 구로구 시흥3동 976-6 삼미주택 나-301
이은열 대구시 동구 신암4동 20번지 동서타운 3동 211호
최창수 대구시 동구 각산동 896
최태룡 경남 울산시 중구 학산동 17-7 20/4
피동진 인천시 남동구 간석1동 268-16 4/3

● 디스켓 박스(10장들이) ●

김정도 부산시 북구 화영동 주공아파트 16동 207호
박기철 서울 성동구 옥수동 527-2 15/8
맹지영 대전시 서구 복수동 283-294
엄호성 대전시 중구 유천2동 214-6
호종화 서울 성동구 자양2동 651번지 25/1

● 컴퓨터 용어사전 ●

강진석 서울 강서구 내발산동 705번지 이화연립 4동 104호
국인숙 경기도 안양시 비산동 주공2단지 복합상가내 108호
김태용 서울 마포구 아현3동 614-1 10/1
박일수 서울 용산구 용산동 2가 1-362
홍성룡 서울 성북구 석관1동 326-15 26/1

● 디스켓 자켓(4장들이) ●

김경기 서울 성동구 사근동 226-40 5/2
김평빈 광주시 북구 운암동 대우아파트 다동 502호
선상기 대전시 중구 중촌동 399-11 9/2
이동련 서울 성동구 중곡1동 234-19
이정훈 서울 중랑구 묵1동 128-34
정상형 경기도 화성군 태안읍 안녕리 산 7-64
채석룡 부산시 북구 엄궁동 엄궁아파트 10동 203호
채희선 서울 도봉구 쌍문3동 103-144 15/10
최문석 전남 여수시 공화동 625-3 9/5
황재현 경남 진주시 상평서동 199-4 진주아파트 107

● IBM PC 입문 ●

김용순 서울 은평구 응암1동 42-78 13/9
서광석 충남 천원군 성환읍 성월리 144 성환주택 가동 102호
신태선 서울 노원구 상계동 주공아파트 408동 1501호
장종철 광주시 북구 용봉동 157-33
최종두 충북 청주시 우암동 386-1

※ 서울 거주자는 2월28일까지 신분증을 지참하시고 직접 본사로 나오시면 해당 상품을 드리며, 지방에 계신 분은 우송해 드리겠읍니다.
기간은 2월28일까지이며, 이후는 무효로 처리합니다.

해인약국

취재/이향선 기자

머리가 아프고 열이 나면 의례히 찾아가는 곳이 약국이다. 그러나 오전이나 오후에 찾아가면 손님이 번잡하기 일쑤이다. 이럴 때 약사는 일일이 손님에 대한 기록을 해야 하고 손님은 자신의 약을 조제할 시간을 고통스럽게 기다리는 번거로움이 있다. 이러한 불편함을 컴퓨터란 해결사로 대처하고 있는 약국이 있다하여 찾아가 보았다.

컴퓨터란 아주 편리한 도구임을 알게 된 약사

구로동의 해인약국(약사:박순철, 김영순)의 문을 열면 잘 정돈된 약품상자와 여러종류의 약사이로 얌전히 앉아있는 16비트 컴퓨터와 인사하게 된다.

기자가 마침 이곳을 찾았을 때는 약사가 손님의 처방기록을 작성하고 있던 중이었다. 처방전을 받은 손님은 매우 만족스럽게 약사의 친절한 인사를 뒤로 약국문을 나선다.

해인약국의 약사 김영순씨가 컴퓨터를 처음 접하게 된 것은 약사 임상 생약 연구회의 회원이었던 87년 6월이었다. 이 연구회의 한방에 관한 강좌에서 약국에서의 컴퓨터의 이용에 관한 강의를 듣게 되었고, 하루에도 수많은 기록과 처방이 행해지는 약국에서도 다양한 환자관리 및 환자 차트관리, 고객관리와 처방, 거래처관리와 약품관리를 할 때 컴퓨터를 이용하면 좀더 효율적인 업무를 운영할 수 있으며 의료지식의 전반에 관한 정보를 약사들이 서로 편리하게 교환할 수 있다는 인식이 약사들 사이에서 점차 두터운 공감대를 형성하게 되자 김영순씨 자신도 컴퓨터의 편리함과 필요성에 관심을 가지게 되었다고.

그뒤 88년 7월 지금의 장소에 해인약국을 개업하면서 현재의 컴퓨터 시스템을 구입하여 본격적으로 자신의 약국업무에 컴퓨터를 이용하게 되었다고 한다.

컴퓨터를 갖추고 나서 처음으로 활용하였던 것은 체질과 투약 및 치료효과에 관한 기본 병력, 부작용과 관련한 고객관리와 위장병이나 당뇨와 같은 질병에 관한 정보의 이용과 다양한 약품에 관한 관리 및 각각의 거래처에 관한 처리였다.

약국 관리프로그램 파파스

해인약국에서 사용하고 있는 관리 프로그램은 파파스(PAPAS)로서 약국의 일상업무인 고객관리를 세대별로 관리하도록 되어 있고, 대화형 메뉴방식을 이용하여 일종의 전문가 시스템과 같이 키를 선택하면 지시하는 화면이 나오도록 하여 간편하게 사용할 수 있게 되어 있다. 또한 판매관리 및 재고, 구매, 경비지출과 세무지출에 관한 사항을 처리할 수 있게 한 것이 큰 특징이다.

이 소프트웨어의 구성은 고객관리, 판매관리, 양방 조제관리와 한방 처방을 처리하는 프로그램 등이 다양한 메뉴 선택방식으로 구성되어 있다.

화면 1은 파파스의 메인 메뉴를 나타내고 있다.

●하드웨어와 OS

파파스를 사용하려면 16비트 컴퓨터 XT나 AT가 있어야 하고 20MB 이상의 하드디스크를 가지고 있어야 하며, 1대의 플로피디스크 드라이브와 24핀 프린터가 있어야 한다.

사용되는 OS는 DOS이고 작성에 사용된 프로그램언어는 RM코볼과 어셈블리어이다. RM코볼은 버전 2.1을 이용하여 주프로그램을 작성하였고, 주프로그램에서 호출하여 처리하는 서브프로그램은 어셈블러 버전 4.0을 이용하여 작성하였다.

●고객관리

고객관리는 고객이 래국시에 이름, 전화번호, 세대별명, 주소중에 한가지를 입력하여 검색하도록 하며, 만약 기 등록된 고객이면 그 내용이 디스플레이되고 등록되지 않은 고객일 경우는 신규등록을 하도록 되어있어 전화번호와 세대별명을 입력하도록 한다.

그리고 등록된 세대중에 가족중의 일부가 변경되거나 이사한 세대가 있어서 등록에 관하여 수정할 필요가 있을 때 삭제하거나 변경이 가능하며, 이에 대한 내용을 인쇄할 수 있도록 하였으며, 래국한 날에 대한 양력과 음력의 환산기능과 바이오리듬도 이용할 수 있다.

고객관리의 또다른 특징은 고객의 주변의 가족, 친지, 동료의 인적사항을 입력시킨 후 성명과 단체, 출신도, 성별, 연령, 직종, 종교, 취미, 출신교 등을 조회할 수 있도록 한 서비스 프로그램인 인명록과 이미 입력된 고객이나 거래처 및 기타 여러사람에게 자료 및 청구서 등 DM을 발송하거나 축하카드를 보내거나 할 때 사용하도록 한 DM 자료현황을 조회할 수 있다는 것이다.

그리고 래국/조제/판매에서 입력된 세대를 인쇄하여 보관할 수 있는 고객

카드의 작성이 가능하다.

●양방조제관리

약을 조제할 때는 조제현황에서 고객의 이름, 전화번호, 세대명, 주소 등으로 고객에 관한 사항을 찾아서 조제한 사항중 최근 조제분부터 조제현황을 참조할 수 있다. 그리고 조제내역에 복사, 조회, 수정, 삭제 등을 필요로 할 때마다 변경하도록 되어 있다.

조제현황은 우선 4가지의 최근 조제내역을 나타내 준다. 이렇게 하여 전 조제 사항을 조회한 다음 조제를 입력하고 처방전을 내리도록 한다.

약사가 처방을 내리기 전에 처방하고자 하는 질병에 관하여 조회하려면 질병 정보관리 기능을 통하여 질병의 번호나 질병명을 입력하여 할 수도 있다.

여러가지 질병에 관한 정보는 질병 계통명을 수정하거나 추가가 있어서 수정과 그 내용을 복사할 수 있고 신규 질병에 관한 입력도 할 수 있다.

또한 래국한 사람에 대하여 매약, 폐기, 삭제, 반품, 이동에 관한 사항을 변경시킬 수 있고 판매처리가 되면 조제등록을 하지 않고도 조제봉투를 인쇄할 수도 있으며 간단한 기록도 할 수 있다.

한가지 특이한 것은 표준 처방 관리가 마련되어 있어 조제시에 표준처방(TP)을 이용할 수 있고 표준 처방의 내용을 등록, 조회, 수정, 삭제할 수 있도록 하였다.

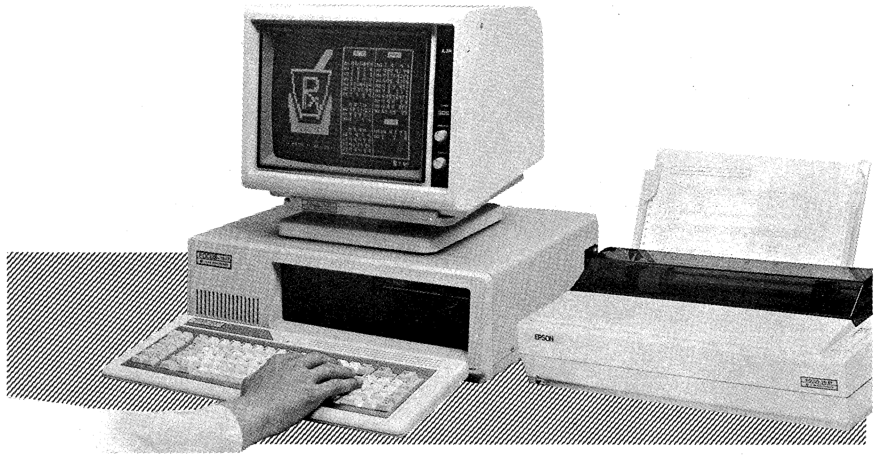
그리고 표준처방 현황에 관하여는 등록되어 있는 모든 표준처방이 코드순으로 나타나기 때문에 파악이 용이하다.

●판매관리

이 부분은 약품명과 약품별 코드와 제약회사별 및 거래처에 관한 관리와 현재 판매상황, 재고상태, 자금수지관리 등에 관한 내용이다.

약품은 약품명과 약품 코드별로 등록을 할 수 있으며, 약품의 주문과 입고에 관하여는 등록시킨 모든 약품중에서 재고가 미달되는 약품만 화면에 나타나고 주문한 약품만을 입고 시킬 수 있으며, 삽입된 약품을 입고 시킬 수도 있다. 그리고 약국과 거래하는 모든 제약회사, 도매상 등의 원장의 관리 및 각 거래처의 등록에 대한 정보를 관리하도록 되어 있다.

〔화면 1〕 파파스 메인 메뉴 화면



자금수지관리는 수입, 지출을 가계부 형식으로 관리할 수 있게 되어 있으며, 등록되어 있는 거래처의 이름 및 색인에 관한 것을 인쇄할 수 있도록 되어 있다.

판매조회 및 재고조사를 통하여서는 판매된 모든 약품과 조제실적, 약품의 폐기, 반품, 이동과 삭제 등의 원가계산과 실판매, 판매원가, 판매수익을 연보, 월보, 일보로 조회하여 볼 수 있고 인쇄할 수 있다.

또한 모든 약품을 판매된 즉시 입력시키기란 약국의 업무에서는 쉬운 일이 아니다. 그러므로 약이 어느 정도 판매되고, 또한 재고량에 대하여는 일정주기마다 재고를 조사한 후 현 재고량을 입력시키면 현 재고량의 차이만큼을 판매가 발생된 것으로 처리한다.

그리하여 등록되어 있는 약품중에서 재고에 관한 사항과 제약회사별 재고액과 총 재고액 등의 관리를 처리한다.

또한 표준 소매가에 관한 정보를 통하여 국내의 각 제약회사에서 생산되는 모든 약품의 포장단위, 소매가, 제약회사 등을 입력시켜 조회할 수 있도록 되어 있다.

그 외에 약국의 주소나 전화번호, 공용의 패스워드, 대표 패스워드를 변경할 수 있도록 하는 기능도 있다.

●한방

이 프로그램은 한방 처방에 관한 것으로 한방의 처방이 양방의 제조 형태와는 다르다는 것을 토대로 이루어 졌다고 볼 수 있다.

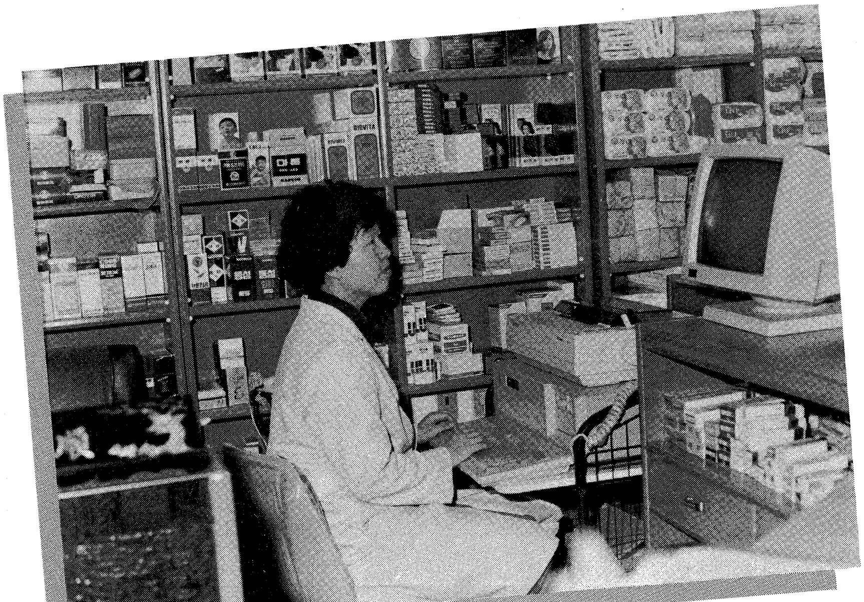
한방은 체질에 따라 나타나는 증상이 다양하므로 그에 대한 여러가지 경우인 260종류의 문진을 실시하여 그 문진에 대하여 300종류의 탕에 대해 처방을 선택한다. 그리고 그에 관한 정보를 조회할 수 있도록 되어 있다.

또한 지역적으로 멀리 떨어져 있는 사람에 대한 문진을 실시할 수 있도록 문진표를 작성하여 인쇄할 수 있다.

컴퓨터 약사의 소망

파파스는 아직 완벽한 약국관리용 소프트웨어는 아니며 한방과 같이 질병이나 증세를 입력하면 그에 대한 처방을 내리는 소프트웨어가 양방조제에서도 보강되어야 한다.

즉, 양방의 조제나 처방시에도 질병



많은 사람들이 건강한 생활을 할 수 있도록
노력하는 컴퓨터 약사가 되는 것이 소망이라고.

과 증세의 입력만으로 자세한 처방전과 병명이 나올 수 있어야 하며 그래픽 기능—예를 들면 위장병 환자가 래국시에 질병에 관하여 도표나 위의 실질적인 형태를 화면상에 나타나게 하여 환자의 상태를 알려주게 하는 방법—이 보강되어 다양한 용도로 쓰일 수 있다면 더욱 강력한 관리 프로그램이 될 수 있다.

김영순약사는 맨처음 컴퓨터를 이용할 때는 익숙하지 못해 약간의 불편함이 따랐지만 현재는 고객관리와 거래처관리 및 자금수지관리에 많은 도움을 받고 있으며 잘 알려지지 않은 정보나 전반적인 의료지식을 교환하는 등 이 프로그램을 유용하게 쓰고 있으며 특히 고객들을 처방할 때 즉각적이기 보다는 자세한 사항을 충분히 조회한 후에 실시하여 고객에게 신뢰감을 주고 있으며 래국하는 고객들도 반응이 좋다고 한다.

그러나 아직도 어려운 점이 있다면 컴퓨터를 이용할 수 있는 시간에 문제가 있다고 한다. 특히 오전이나 저녁시간에는 많은 고객이 래국하기 때문에 현재의 컴퓨터 1대로 처리하는 것이 번거롭고 불편할 때가 있다.

그래서 시간상의 문제점을 극복할 수 있도록 컴퓨터 활용에 익숙해 지는 것이 우선 과제라고 한다. 그리고 현재 사용하고 있는 약국용 관리 프로그램 중 표준처방에 관한 정보에 관한 조회를 할 때 천차만별인 환자 각각의 경우에 모두 적용할 수 있는 다양한 정보조회가 필요하다고 한다.

컴퓨터에 관하여는 초보자의 단계라고 하는 김영순약사는 컴퓨터 이용을 단지 약국의 관리 프로그램이라는 측면에서 이용하는데 그치지 않고 고객관리에서 DM 발송시에 그때 그때마다 질병에 관한 자료를 발송하여 각종 질병에 관한 예방을 숙지시키거나 전염병 예방을 공지하여 미연에 방지할 수 있도록 하여 공중위생에 보탬이 되었으면 하는 소망을 가지고 있다.

그리고 폭넓은 컴퓨터 이용자로서 컴퓨터를 사용하는 약사들 끼리도 PC를 이용한 통신으로 의료지식에 대한 정보를 교환하여 보다 효율적인 약국업무에 임할 수 있는 방안을 기대하며, 그러기 위해 컴퓨터에 대해 계속 공부하려는 옹골찬 포부를 가지고 있다.

현재 해인약국에서 사용하는 파파스를 이용하는 약국은 150여개 정도되며 파파스 이외에도 다른 소프트웨어로 컴퓨터를 이용하고 있는 약국들도 상당수가 되고 있다.

이와 같은 추세이고 보면 작은 중소기업이라고 일컬어 지는 약국의 전반적인 전산화는 비교적 빠른 시일에 이룩되리라 기대된다.

이제는 머리가 아프고 기침이 나고, 열이 나서 약국을 찾아가게 되면 약의 처방에 의심하지 않고 자신의 증세와 질병에 관하여 정확히 알 수 있으며 이러한 신뢰감이 컴퓨터와 약사, 그리고 고객사이에서 증대됨으로써 더욱 더 건강한 생활을 영위할 수 있게 된 것이다.

월간 퍼스널컴퓨터의 정기구독자에게는 아래와 같은 특전이 마련되어 있습니다.

—정기구독자는 본사가 보유한 각종 정보 및 자료를 무료로 배부받으실 수 있읍니다.
—본지에 게재되는 각종 프로그램을 디스켓에 담아 무료로 보내드립니다.
—본지 및 본지의 자매지(월간 전자과학) 주최로 개최되는 세미나 및 각종 행사에 무료로 참석하실 수 있읍니다.
—본지에서 행하는 각종 부대사업에 할인혜택이 주어집니다.
—구독기간중 책 가격이 인상되더라도 종전가격으로 구독이 가능합니다.

●구독신청은

—시중은행에서 지로용지를 이용하시거나 우체국에서 대체구좌 또는 소액환으로 송금하시면 됩니다.
은행 지로번호 : 3000667
우체국 대체구좌번호 : 010256—31—0526772

●정기구독료는

—1 년 구독시 30,000원
—6개월 구독시 15,000원

●구독신청시 주의사항은

—구독기간과 주소(우편번호), 전화번호와 받으실 분의 성명을 정확히 기입하시기 바랍니다.
—배달사고나 기타 변동사항은 본지 영업부로 즉시 연락하시기 바랍니다. 신고전의 분실사고는 책임지지 않습니다.

●정기구독 문의처는

—서울시 용산구 갈월동 17 ㊄ 140-150
월간 퍼스널컴퓨터 영업부
전 화 : (02)754—1044(대표)
FAX : 757—1049

■ 전국총판 ■

●서울●	종로, 중구	명 문 사	774—6314
	성동, 동대문	홍익도서	454—1810
	성북, 도봉	진 원 사	981—4884
	강남, 강동	광 일 사	415—7330
	관악, 동작	관 악 사	889—4504
	서대문, 은평	형 제 사	304—9462
	영등포, 구로	영 도 사	613—5381
		진명서점	272—5871
		관음서적	266—7792
●부산●		종합기술서적	462—2431
		영광도서전시관	805—9500
		북부총판	805—9677
		남부총판	756—2727
●대구●		한일서적	23—7101
		학 원 사	22—0336
●인천●		동양서점	73—3384
●광주●		삼복서점	22—0258
●경기●	부천	대경서점	62—3672
	안양, 안산	사랑글방	57—1916
	수원,	동아서점	5—6436
	성남	영광서점	2—6532
●강원●	원주	원주종로서적	42—2943
	삼척	영 동 사	72—6060
	강릉	삼 일 사	2—2314
	춘천	대진서점	54—5148
●충북●	충주	문 학 사	2—2814
	청주	화 문 당	55—0503
●충남●	천안	동방서적공사	63—3314
	대전	동양서점	22—1538
●경북●	포항	경북서점	2—2717
	구미	중앙서점	52—7915
●경남●	진주	동아서점	42—3768
	진해	학애서점	2—2523
	울산	우정서점	43—6003
	마산	기술서적	56—1417
●전북●	전주	홍지서점	6—5311
	군산	동원서점	42—3776
●전남●	목포	문 화 당	5—3086
	여수	대양서점	62—2111
●제주●		광양서적	53—5077

책을 마감하고

●새롭게, 꾸준히, 적극적으로…

그러나 아직 미숙합니다. 그것도 아주 많이 미숙합니다. 배울 것이 천지에 가득합니다. 그러나 이럴 때일수록 당황하지 않는 침착한 마음이 필요하다는 것을 압니다. 단번에 만리장성을 쌓을 수는 없읍니다. 만약 쌓을 수 있다해도 그것이 허술한 장성이라면 쌓기를 거부합니다. 아주 느리게 쌓아지더라도 예전에 범했던 실수를 재차 반복하기엔 우리네 인생이 너무도 짧습니다. 열심히 그러나 조급하지 않게, 진득하니 앞으로 10, 20년 아니 40년 후에 돌이켜 봤을 때 후회되지 않는 젊은날로 가꾸렵니다. 〈賢〉

●셋!

학창 시절에 인공지능분과에 있었다.

분과 두목은 보기만 해도 공포분위기를 자아내던 미혜!

사방 팔방으로 뒤엉켜진 그녀의 머리카락만 보아도 두목다운 기질이 충분히 엿보인다. 언제 시커먼 선글라스라도 사다 드리면… 폼이 근사할텐데…히히 〈榮〉

●많은 것을 알아 간다는 것은 그만큼의 고통을 참아내야 한다는 것인가 봅니다. 우리들을 가두고 있는 마음의 구속들을 홀가분하게 벗을 수만 있다면 오죽 좋겠읍니까.

아픔으로 다가섰던 모든 것을 기쁨으로 바꿀 수 있는 커다란 용기를 얻을 수 있다면 아쉬움없이 모든 것을 내어 놓겠읍니다.

가장 선량한 마음으로 서 있었다는 것이 거짓이 아니었으며, 정말로 사랑하였다는 것이 후일에 가서도 부끄럼없이 말할 수 있을 때 사랑을 되찾을 수 있다는 희망을 버리지 않고 살아 가겠읍니다.

오래전부터 보석을 지녔다고 해서 그것이 사람을 행복하게 하는 것은 아니기에 나의 사랑하는 친구를 위하여 아낌없이 주고 싶습니다. 그렇게 하는 것이 아름다운 모습이기에 매일의 고통 속에서도 회생할 수 있을 겁니다. 그리고 나를 너무나 사랑한 부끄러운 내게 신의 은총이 아직까지도 남아 있다면 전과 같이 아니 그전보다 더욱더 그를 사랑하렵니다. 〈둘리〉

●태초에 하나님께서 6일째 사람을 만드시고 7일째 돌로 CPU되라 하시매 Z80이 생기고 Z80은 자가 수정하여 8086을 낳고 8086은 8088을 낳고 8088은 80286을 낳고 80286은 80386을 낳고 이런 전차로 하여 인간들이 CPU와 친하지 못하여 나를 보내어 CPU를 다스리라 하였건만 CPU는 어디가고 원고지만 남았는가? 담아야할 내용은 많은데 메모지 용량은 작고 메모지는 에러가 속출하니 답답하구나 〈石心〉

●내게 주어지는 모든 시간들을 가슴으로 맞이하고 싶다.

때로는

절망과 강한 회오리 바람일지라도

삶의 현장속에서 넓은 가슴을 지닌

아량으로 포용하고 싶다.

그러면서 나는 좀더 깊이있는 인생을

살겠지…. 〈淑〉

●또 한권의 책을 만들었다.

이 한권의 책을 만들기 위해 나의 사생활을 침해 받으면서 정신적·육체적 피곤이 쌓이고, 입술은 부르트고 속이 쓰려 병원에 한번 가야겠는데 갈 시간적 여유조차 주질 않는다.

언제나 이런 상태에서 벗어나게 될런지? 〈哲〉

●뭣도 모르고 뛰어들어 코피깨지고

진창 두들겨 맞은 촌놈처럼 허덕이다 보니

한달이 휭-가고

이런 자유스러움도 기웃거릴 순간이 올줄이야

굴레에서 벗어났다.

기지개를 펴고 나아가자! 했더니

왠걸, 더 지독한 놈이 가로막고 있을 줄이야

어떻든, 지난 1개월은 내게 색다른 세상으로 다가왔고, 신바람나 천방지축으로 날뛰는 고귀한 시간이었다.

여기까지 읽어준 애독자들이여!

새싹이 푸릇푸릇 돋는 3월에 더욱 알찬 모습으로 기다리겠읍니다. 〈鎔〉

●기차를 탔지요.

스치는 창밖의 풍경

고스톱에 심취해있는 아저씨들

칭얼대는 아이들

수선스런 분위기

이들이 정겨움으로 다가왔지요.

처음본 이와의 대화에서도

친숙함을 느껴보았지요.

그 이유는…

P.C 친구들

이 겨울이 가기전 떠나보시지

않으렵니까!?

호올로. 〈美〉